Optimised Task Allocation Using Dynamic Production Data in Human-Robot Teams

Thomas Anthony Smith

Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy

June 2020

Abstract

The demand of both industrial and consumer customers for increasingly higher degrees of customisation in products will see greater amounts of high mix production in the future of manufacturing. Despite this, automation must be implemented to improve the efficiency and output of manufacturing processes. However, traditional automation methods are often unsuitable due to long lead times for setup and little flexibility to adapt them to new tasks. Human-Robot (HR) teams provide a potential way to implement easily reconfigurable automation into future factories by utilising the best characteristics of human workers such as adaptability and intelligence with those of robot workers such as strength and repeatability. Robust task planning is required to implement such HR teams. However, current approaches allow adaptation to change in performance or composition of HR teams or optimisation of tasks as a whole but not necessarily both.

In this research, a novel generalised task planning framework is proposed that uses a semi-online task planning approach, utilising online production data to determine worker capabilities then planning a manufacturing task for the HR team offline between task iterations. A system architecture is defined for such a framework but the focus of this research is the development and testing of the core technologies required for the framework to function to assess its utility. These include dynamic cost functions utilising online production data to accurately quantify the capabilities of human and robot workers across a work shift. These use continuous variables to quantify gradual changes in worker performance across a work shift; and discrete variables to detect instantaneous changes in capabilities that occur during a single task iteration. Additionally, a dynamic task planner is developed that implements dual layers of the Discrete Gravitational Search Algorithm to search for an optimum set of task assignments and task plan for a HR team given worker costs. Finally, mechanisms are proposed to intelligently implement task replanning across a work shift to optimise a HR team's performance whilst ensuring it does not occur too frequently or unnecessarily.

These core technologies were tested individually in example cases then combined together to test the ability of the task planning framework to optimise the performance of a HR team in two example manufacturing tasks across simulated work shifts. This showed that the dynamic cost functions represent an effective way to quantify and detect any changes in a worker's capabilities across a work shift. Additionally, task replanning was shown to improve the performance of the HR team in some scenarios, such as the human worker being over fatigued, by reassigning subtasks to the robot worker as their performance declines. Importantly, the proposed task planning framework represents a generalised methodology that can easily be redeployed to different manufacturing tasks or compositions of HR teams.

Acknowledgements

I would like to thank a number of people whose support was instrumental in allowing me to finish this thesis.

First of all, I would like to thank my supervisors David Branson, Atanas Popov and Panorios Benardos for their invaluable guidance and support throughout the course of my studies.

I would also like to thank all my colleagues in the Advanced Manufacturing Building at the University of Nottingham whose kindness and encouragement made it a pleasurable working environment.

Additionally, I would like to thank the members of the Chatty Factories project across all of the institutions involved that were also a pleasure to work with.

Finally, but most importantly, I would like to thank my Mother and Father for their encouragement, love and unfailing belief in me.

Table of Contents

Nomenclature

Symbol	Description
A where $A = [\alpha_1, \dots, \alpha_N]$	A potential set of task assignments
B_i	The i^{th} simplified task plan
$C_{i,j}$	Cost for a worker *j* to complete a subtask *i*
D_j	Average number of task iterations completed in assignment period and subtask *j* required for Digiesi's fatigue model
$E_{i,j}$	Expected completion time for subtask *j* at task iteration *i* from Digiesi's Fatigue model
F_i	The fitness value for the i^{th} searcher agent in DGSA Layers 1 & 2
$G(\delta)$	Gravitational constant in iteration δ of the main phase of DGSA
H_j	Work element time for manufacturing subtask *j*
I where $I = [1, \dots, N]$	The assembly plan
$J_{i,k}$	The idle cost for a worker *k* whilst waiting to complete a subtask *i*
K	The number of best solutions in the DGSA
L	Unique corresponding constraints list for a subassembly
M	Total completion time cost for a manufacturing task
N	Total number of subtasks in an overall manufacturing task
O where $O = [o_1, \dots, o_N]$	Execution constraints list for an assembly plan
P where $P = [p_1, \dots, p_N]$	A potential task plan
Q_i	The i^{th} subassembly of a potential task plan
R	The tolerance region boundary for the fatigue variable defining the maximum or minimum completion times that can be considered insignificant variation from expected completion times
S	Number of searcher agents used in the GSA/DGSA
$T_{j,k}$	Average assignment period length for worker *k* and subtask *j* required for Digiesi's fatigue model

U	Swappable task plan elements
V	Missing subtasks list
W	Number of available workers
X_i	Position of a searcher agent i in the solution space of the DGSA
Z	Corresponding execution constraints list to missing subtasks list
a	The percentage area of the specified sealant line over which sealant is applied
b_i^j	The j^{th} subtask in the i^{th} simplified task plan
c	The mean cost of previous discrete events in the past κ number of task iterations
d	A dimension of a GSA/DGSA solution space
e_j	The maximum acceptable percentage increase or decrease in completion times from the expected completion time for the fatigue variable
$f_{i,j}$	A cost function variable where i gives the identity of a variable for a subtask j
$g_{i,j}$	A cost function variable weighting where i gives the identity of a variable for a subtask j
h_j	The tolerance to natural variation in completion times for subtask j required for the fatigue variable
I	The total completion time of a set of task assignments and task plan
m	A small move in the DGSA solution space
n	The number of dimensions in the GSA/DGSA solution space
o_j	The execution constraints for a subtask j
p_j where $p_j \in I$	A task plan element
q_i^j	The j^{th} subtask in the i^{th} subassembly
$r_{i,j}$	The moving average completion time of a worker for subtask j in the i^{th} task iteration
$t_{i,j}$	The completion time for subtask j in the i^{th} task iteration
u_j	The identifier of a subtask j in the swappable task plan elements list, U
w	Initial task iteration of the current task assignment period for a worker
x_i^d	The position of the i^{th} searcher agent in the d^{th} dimension of the DGSA
z	The angle the applied sealant line deviates from the specified line

Γ_j	The number of iterations of a manufacturing task that would have been completed whilst the human worker has not been assigned the j^{th} subtask
Θ_j	The last task iteration that the j^{th} subtask was assigned to the human worker
Λ	The base set of task assignments
Φ	The cool down modifier function
Ψ	The set of locked in task assignments
Ω	The fatigue variable boundary defining the completion times where the maximum or minimum cost of the variable is reached
α_j where $\alpha_j \in \{1, \dots, W\}$	The j^{th} task assignment in a set of task assignments, A
β	The number of available cost function variables
δ	The current iteration number of the main phase of the DGSA
ε	The number of successfully completed iterations of a subtask since the last discrete event
ζ	The length of the moving average for completion times, $r_{i,j}$
η	The threshold cost that must be exceeded to indicate that a worker's performance is significantly better than other workers and their task assignment should be locked in.
ϑ	The maximum acceptable idle time for workers
ι	The number of occurrences of discrete events in the past κ number of task iterations
κ	The number of past subtask iterations to consider previous discrete events from
λ	The maximum number of iterations used in the main phase of the DGSA
μ	The total number of primitive tasks which a cost function variable influence
ν	The percentage length of the specified line over which sealant is applied
ξ	A constant to attenuate the growth of the frequency modifier function
o	A constant to attenuate the decay of the cool down modifier function
$\rho_{i,k}$	The idle time for worker k whilst waiting to complete a subtask i

ς	The frequency modifier function
σ	The cost of an individual sealant application
τ'_j	The synthetic measure of fatigue for a subtask *j* used by Digiesi's Fatigue model
υ	The output cost of the discrete variable after the last occurrence of a discrete event
φ	The small move operator to execute a small move, *m*, in the DGSA solution space
ψ_j	The locked in identifier for the j^{th} subtask of a potential task plan, *P*
ω	The threshold cost difference to define when a worker's costs have changed significantly enough from those in the last task replanning attempt to execute the current replanning attempt.

Acronyms

Acronym	Full Name
DGSA	Discrete Gravitational Search Algorithm
DML	Dependant Movement Length
DMO	Dependant Move Operator
GSA	Gravitational Search Algorithm
HR	Human-Robot
IML	Independent Movement Length
IMO	Independent Move Operator
Kbest	Set of *K* best agents in DGSA

List of Figures

List of Tables

1. Introduction

1.1. Introduction and Background

The future of manufacturing will see more high mix production as both industrial and consumer customers require a high degree of customisation in the products they order. Despite this, manufacturers still desire to implement automation in their factories to improve the efficiency and output of manufacturing processes. However, the limitations of traditional automation methods prevent their use in these situations due to the traditional need for long setup lead times and reduced adaptability in current static production lines.

Future visions of manufacturing such as Industry 4.0 highlight Human-Robot (HR) teams as a potential way to implement reconfigurable automation into future factories. HR teams utilise the best characteristics of human workers such as their adaptability and intelligence with those of robot workers such as their strength and repeatability. The rapid commercial development of human collaborative robots (cobots) over the past decade has resulted in cobots with higher payload capacities and repeatability that are safe to work closely amongst human workers. This enables HR teams to be realistically considered for manufacturing tasks and has seen large areas of research to develop methodologies to implement such teams.

Although the concept of HR teams provides a potential solution to implement reconfigurable automation into future factories, innovative methodologies are required to implement them in manufacturing tasks in a useful way. Such methodologies must ensure that both robot and human workers are implemented in task that they are best suited to. However, the methodologies should also allow the HR team to be redeployed to another manufacturing task quickly and allow adaptation to changing worker capabilities.

The literature review in Chapter 2 highlights two potential ways to implement robots amongst humans in HR Teams: as assistants to human workers, where the robot indirectly participates in a task by interacting with the human worker, or as their peers, where the robot takes turns with the human worker to directly complete elements of the manufacturing task. Whilst such measures can be helpful for a human worker, it can result in an underutilisation of cobots that have an ever-increasing range of capabilities. Utilising a robot worker as a peer to human workers allows them to instead directly contribute to the completion of manufacturing tasks, fully utilising their capabilities. This methodology is thus chosen to be further explored in this research as it offers a path to partial automation in settings where full

automation of manufacturing tasks is not currently possible. However, to implement HR teams with robots and humans as peers to each other, robust task planning is required to ensure workers are implemented appropriately. To achieve this, task planning must ensure that each task is assigned to the optimal worker for completion based on the capabilities of the worker and the requirements of each task.

The literature review in Chapter 2 also shows that two planning methodologies, online and offline task planning, emerged to plan manufacturing tasks for such HR teams. Online task planning methodologies assign tasks one by one as they are being executed allowing adaptation to change in performance or composition of HR teams but not necessarily optimisation of tasks overall. This is typically achieved using machine vision technologies that observe the shared workspace of the HR team and implement the robot worker on tasks that are available for completion. Offline task planning instead plans entire tasks offline based on static indicators or models of the performance of workers, optimising tasks as a whole but not allowing much adaptability. Task assignments and plans generated are then typically executed by a HR team with limited capabilities to react to collisions or failures allowing the robot worker to stop safely. In other recent research, task planning methodologies have emerged for HR teams that consider offline task planning but utilising online data regarding the current world state of the manufacturing environment or the capabilities of human workers. This has seen very limited use in previous research which developed semi-online task planning methodologies around specific use cases for a defined manufacturing task and HR team.

A knowledge gap is highlighted in the literature review for dynamically planning tasks for HR teams based on the variability in performance and capabilities of workers across a work shift. Semi-online methods of task planning have the potential to allow efficient and adaptable implementation of a HR team in a manufacturing task. However, such approaches have so far focused on task replanning based on specific factors influencing HR teams and the effect of these on the capabilities and performance of the HR team. These existing semi-online task planning methodologies do not consider using multi-variable cost functions, which update a cost for a worker to complete a manufacturing subtask using data from the HR workspace. In addition to this, existing semi-online task planning methodologies do not fully consider the re-assignment of assembly subtasks with the changing capabilities of human workers across a work shift. This knowledge gap can be further refined as the requirement of a generalised methodology to allow for semi-online structured task replanning of a manufacturing task over a work shift for a HR team given the variability of workers capabilities.

In this research, a system architecture is proposed to apply a semi-online task planning methodology across a work shift for a HR team. This defines how online production data is used by multi-variable dynamic cost functions to quantify the capabilities of workers as costs. These costs are then used by an offline task planner to optimise the assignment and sequencing of subtasks of an overall manufacturing task. As the HR team then execute the manufacturing task, online production data is gathered, and the dynamic cost functions must update worker costs based on this new data. Utilising these updated costs, the task should be replanned offline at set intervals based on the changing performance and capabilities of workers.

In this thesis, the core methodologies for this system architecture are developed beginning with the dynamic cost functions required to quantify a worker's capabilities based on online production data. Cost function variables are proposed to enable the use of continuous online data to determine gradual changes in worker performance and discrete data to detect instantaneous changes in their capabilities. In addition to this a dynamic task planner is proposed to optimise both the assignment of subtasks to workers and the sequence in which they must be completed. Finally, techniques are developed to determine when task replanning attempts should occur across a work shift and to determine whether task replanning is necessary.

1.2. Research Aims and Objectives

This work aims to provide a novel generalised methodology to implement robot workers amongst human workers as a part of a HR team whilst optimising implementation of both workers and allowing adaptability. Based on this research aim and the knowledge gaps identified in the literature review in Chapter 2, research objectives are defined given current limitations in quantifying a worker's capabilities using online production data and using this information to replan tasks for HR teams during a work shift.

First, based on the limitations of utilising online production data to quantify the current capabilities of a worker, the following research objectives are defined:

- Formulate dynamic cost functions for human and robot workers, consisting of variables that can use continuous or discrete production data to quantify the capability of each worker to complete each subtask of an overall manufacturing task;
- Develop mechanisms to update the output cost for a cost function variable given online data obtained from the collaborative workspace over iterations of a manufacturing task. These mechanisms must ensure the output cost from a cost function variable accurately quantifies the capabilities of human and robot workers.

Next, given the limitations of task planning in using such online production data the following objectives are defined:

- Produce a task planning methodology to find an optimum set of task assignments and task plan for a HR team given the costs generated for each subtask whilst respecting task constraints and minimising worker idle times;
- Implement mechanisms to ensure that subtasks are assigned to optimal workers if there is a significant difference in worker capabilities;
- Implement mechanisms to trigger task replanning at appropriate intervals but only if worker costs changes indicate this is necessary.

1.3. Contributions to Knowledge

The contributions to knowledge of this thesis are focused around dynamically planning tasks for HR teams based on the variability in performance and capabilities of workers across a work shift.

First, this thesis presents dynamic cost functions that can utilise online production data to update knowledge on the capabilities of workers across a work shift. These dynamic cost functions can react to gradual changes in the performance of workers using continuous variables that analyse continuous online data. The dynamic cost functions can also react to instantaneous changes in the capabilities of workers using discrete variables that react to discrete events in manufacturing subtasks.

Second, this thesis presents a dynamic task planner to optimise both task assignments and task plans for a HR team. This represents a generalised methodology that allows easy adaption to different manufacturing tasks. However, this should also allow different compositions of HR teams containing various numbers of human or robot workers.

Finally, this thesis presents techniques for implementing task replanning across a work shift utilising intelligent methods to trigger when replanning attempts occur and determine if it is necessary to replan tasks to prevent unnecessary computational expense.

1.4. Thesis Outline

Over the course of this thesis the overall research aim is explored to develop a novel task planning methodology to plan manufacturing tasks for HR teams. This begins in Chapter 2 with a thorough literature review into the various methodologies for implementing robot workers amongst human workers in HR teams and identification of potential knowledge gaps. Chapter 3 contains a description of the research methodology beginning with the transformation of identified knowledge gaps into a formal definition of the

research objectives. In addition to this a system architecture is outlined for the task planning methodology proposed in this thesis including definition of the core enabling technologies that are developed in this research to verify the proposed semi-online task planning methodology. Next, Chapters 4 and 5 describe the development of techniques to quantify the changing capabilities of human and robot workers across work shifts through the use of continuous and discrete variables in dynamic cost functions. Chapters 6 and 7 then describe the development of methodologies to utilise costs for workers to generate an optimal set of task assignments and task plan then update them as worker performance changes across a work shift. Finally, Chapter 8 presents the conclusions of this research in addition to possible areas of future research to further develop the methodologies outlined in this thesis.

2. Literature Review

2.1. Introduction

In this chapter a thorough literature review is presented to discuss the works that this thesis builds upon. To provide motivation for this research it is first necessary to discuss the existing uses of automation in industrial manufacturing processes in Section 2.2 to illustrate the need for adaptable Human-Robot (HR) teams for future high mix production. This is followed by a discussion on HR collaboration in Section 2.3 detailing technologies that enable such methodologies, potential roles for robots in HR teams and the levels of collaboration that can be implemented.

Given the description of HR teams it was next necessary to detail how manufacturing tasks are planned to allow HR teams to be implemented. This was broken down into three key areas, beginning in Section 2.4 with a discussion of existing task planning methods within the field of manufacturing. Following this, current research trends in task planning for HR teams are discussed with offline and online task planning methodologies detailed in Sections 2.5 and 2.6, respectively. Finally, Section 2.7 concludes with a discussion of the knowledge gaps in task planning for HR teams highlighted from this literature review that will be investigated within this thesis.

2.2. Existing Use of Automation in Industry

Automation has been implemented in industrial manufacturing for several decades and can benefit manufacturers by lowering the unit cost to manufacture products in addition to increasing the speed of production. Over this time the conceptual methods and approach to implementing industrial robots has, however, changed very little. Industrial robots are typically implemented in highly repetitive simple tasks with minimal variation in the task specification such as painting, welding or manipulation of production components (Hägele et al., 2016). Such simple tasks are achieved by following predefined trajectories with an end effector over high numbers of task iterations in the production lifecycle of a product.

As a result of this type of implementation, large scale industrial robots typically do not include sensors suites that make them aware of changes in their environment and are highly susceptible to colliding with unexpected objects in their workspace. The large size and mass of these robots combined with their fast movement speed can result in lethal levels of force being imparted on human workers that enter their workspace. This has necessitated that these robots operate in isolated environments that are rigorously enforced by interlocked cages to ensure robots cannot operate whilst human workers are present in their workspace. The environment and safety

constraints of industrial robots offer little flexibility in use once installed and require a large setup time when initially implemented or reconfigured for a new production process. This setup typically takes place during the commissioning of an assembly line in a factory and involves the robot being programmed to complete their tasks by specialist engineers. Further tuning of the robot's tasks are often required when production starts over a large period of time.

A current example of this type of automation is in the automotive industry where robots are installed on an assembly line to perform tasks such as spot welding or spray painting the body shell of a car. The robot can be implemented to perform this task repetitively over the lifecycle of the car which could last a few years with minimal variation in the task the robot is performing. This type of automation is suited to large scale production such as this since the time and cost required to implement robots in these tasks can be factored into the commissioning of a factory during the development of a new car model. Although this type of automation is well suited to large scale manufacturing, it is poorly suited to low volume high mix production used in many smaller companies where the difficulty in reconfiguring the robots would make their implementation uneconomical.

2.3. Human-Robot Collaboration

2.3.1. Breaking Down Fences

Research has identified HR collaboration as a solution to implementing automation into manufacturing sectors where this was previously not possible. HR collaboration consists of using the best characteristics of robots such as their strength and repeatability with those of a human such as their adaptability and intelligence. New opportunities for close collaborative work between humans and robots were only made possible by the introduction of human collaborative robots (cobots).

Cobots were first proposed in the 1990s in (Colgate, Wannasuphoprasit and Peshkin, 1996) which recognised the need for closer collaborative work between humans and robots within manufacturing environments. Cobots became more widely commercially available with the introduction of the single arm Universal Robots UR5 in 2008 which has since seen many applications in industrial settings. The availability of robots such as the Rethink Robotics Baxter (Cremer, Mastromoro and Popa, 2016), a dual armed robot which was widely purchased by universities, resulted in a significant increase in the levels of research into HR collaboration.

Typical cobots include several safety systems allowing humans to work in close proximity whilst maintaining their safety. Such safety measures vary widely in operating methodology using both software and hardware-based

measures, but all typically ensure a cobot can detect a collision with a human worker and stop quickly whilst limiting the imparted force to prevent injuries. The safety of cobots allow them to be implemented without interlocked cages amongst human workers. Their size and weight also allow them to be mounted on movable benches, requiring a much smaller footprint than traditional industrial robots and allowing easy relocation within a workspace. Cobots can also be reprogrammed on the factory floor through lead through programming, allowing them to be reconfigured quickly for new tasks by a worker physically moving through positions of a desired trajectory. The disadvantage of typical cobots is that their lifting capacity, which can range from 0.5kg to 16kg, is much smaller than traditional industrial robots which have lift capacities in the order of hundreds of kilograms. As a result of this, research has also now attempted to retrofit existing industrial robots with sensors to allow human workers to work safely in close proximity (Bdiwi, Pfeifer and Sterzing, 2017; Lasota, Rossano and Shah, 2014).

Despite the possibilities cobots offer through HR collaboration, they are typically implemented alone completing the same task repetitively in industry (Salunkhe et al., 2019). HR collaboration has been explored much more in research with the emergence of two different roles for robots in the future of manufacturing. Such roles include viewing robots as assistants to human workers or as peers working alongside them on manufacturing tasks.

2.3.2. Robot Assistants

The use of robots as assistants to human workers has emerged as a strong theme in current research with the emergence of cobots as commercial products. This form of HR collaboration revolves around the concept of a robot only completing assistive actions for a human worker rather than directly engaging in the manufacturing task. In this role the robot performs actions such as handing tools or components to the human worker that they will need during the manufacturing task to enhance the efficiency of their work (el Makrini et al., 2017). Additionally, if components are cumbersome or require support whilst a human worker completes tasks upon them, the robot can perform actions such as holding the component in place to assist the human worker (Goto, Miura and Sugiyama, 2013).

Research in this field is heavily reliant on the observation of a human worker and the workspace to allow a robot worker to recognise and anticipate a human worker's need for assistance in performing a task or detect a direct request for assistance. Technologies required to achieve this can often pose difficult research problems in themselves. It has been shown in previous research such as (Chan et al., 2015) that simple tasks such as object handovers prove difficult requiring a robot to be very aware of its environment and objects within it to accomplish a successful handover of an object. This includes

ensuring an object is grasped to ensure a human can accept it without danger, recognising the location of a human hand and recognising when to release the grasp on an object. These actions are often highly dependent on sensors monitoring a human worker with algorithms such as those for pose estimation being computationally expensive (Asfour et al., 2018).

The recognition of the intended actions of a human worker or the task they are currently completing pose similar research problems which often require similar solutions to work successfully. Through observation of the workspace and the human worker, a robot can determine which step of a manufacturing task is taking place and enact a pre-programmed response to assist the human worker. Previous research such as (Goto, Miura and Sugiyama, 2013) has shown the possibilities and benefits of such capabilities to aid HR collaboration in the assembly of a table. Here a human worker presented a tabletop to a robot worker which located grasping points and held the component whilst the human worker attached the table legs. Additionally, the robot was able to determine when to release its grasp so the human could rotate the table to access additional mounting points and push table legs to the human worker if they were out of reaching distance. Similar assistive actions were implemented in (Hawkins et al., 2013) where a human worker must assemble a specified model from components located in bins, but not all bins are within reach of the human worker. Here the robot worker was required to predict which components the human worker would require next, enabling the removal of bins that were no longer required and their replacement with bins containing necessary components.

Another attempt at such collaboration was shown in (Asfour et al., 2018) which developed a highly capable mobile dual armed cobot to assist human workers that was capable of anticipating the need of assistance to a human worker in addition to responding to a direct request of help. This was achieved through visual detection utilising pose estimation and activity recognition to autonomously determine if assistance was required or natural language processing to perceive requests of help from a human worker. Additionally, the robot was capable of altering its actions during a collaborative carrying task by monitoring the force distribution in its hands and regrasping the object in another position to optimise the provided support. Testing of the system in (Busch et al., 2019) showed that in a factory environment the human worker had to open and close a microphone to communicate with the robot to prevent ambient noise from being misinterpreted as communication attempts. Testing also showed multiple failure events were seen in the regrasping of objects, where the robot didn't grasp the object, and in voice recognition, where the user used the wrong command, or the command was not detected correctly. This highlights a potential problem with this type of HR collaboration as the high dependency of the use of sensors for the robot to successfully provide assistance to the robot worker.

In other research, this methodology has further evolved to allow a robot worker to predict how best to provide assistive actions to a human worker instead of providing a reactionary response. This was seen in (Grigore et al., 2018) where a hidden Markov model was trained from a simulated training set of human workers assembling a chair to predict which supportive measures to provide to a human worker considering their preferences for assistance. The model was then applied to predict which supportive actions to apply to a human worker when assembling a chair through observation of the worker.

The research presented showed that this type of HR collaboration can be beneficial to human workers in tasks that require components to be supported during assembly actions. Such collaboration allowed the human worker to use both hands to complete their tasks, increasing their dexterity compared to having to support components and complete tasks with one hand. It was also shown to be useful in tasks involving large objects by reducing the strain on the human worker when carrying them. Importantly, it was shown that through the use of sensors the robot worker could anticipate the needs of the human worker autonomously and provide assistance through a preprogramed response. This was beneficial as it allowed the human worker to intuitively work with the robot worker as they would with another human. The disadvantage of such methods is that they are highly dependent on sensors for the completion of the manufacturing task meaning that if the sensor's view is occluded it may not be able to detect the human worker or objects, preventing the robot from responding to the human worker. Additionally, supportive measures requiring vocal commands from a human worker were shown to be difficult due to the noise in factory environments and the variability of vocal commands from workers.

Despite these benefits to the human worker, utilising some cobots or larger industrial robots in this way can result in underutilisation of their capabilities in the manufacturing process. Cobots such as the Universal Robots UR10 have a high repeatability of ±0.01mm and lift capacity of 10Kg meaning that in some tasks they may prove as capable as human workers. This shows the underutilisation of the UR10 in (Hawkins et al., 2013) as with the correct programming and sensors the robot is capable of much more complex tasks than passing bins of components to a human worker. This type of HR collaboration would instead be better suited to other cobots such as the Rethink Robotics Baxter due to its lower repeatability of ±5mm (Cremer, Mastromoro and Popa, 2016) which limits its capabilities in manufacturing tasks. Additionally, the utility of this type of HR collaboration is limited to certain tasks involving heavy objects or those that require support during assembly and may be of limited utility in other situations. Due to this, such implementations of robot workers are not considered in this research as highly capable robot workers can be better implemented by direct use on manufacturing tasks to achieve a more efficient use of a HR team.

2.3.3. Robots as Peers to Human Workers

Using robots as peers to human workers is the other strong theme of current research in implementing HR collaboration into industrial manufacturing processes. In such roles, a robot would engage directly in the completion of a manufacturing task alongside a human worker by completing subtasks on work pieces themselves. This type of implementation utilises the benefits of HR collaboration by assigning repetitive and tedious tasks to a robot worker, leaving a human worker free to complete high value-added tasks requiring high skill or experience levels.

To implement robot workers amongst human workers in this way, it is necessary to understand the capabilities of both human and robot workers to determine their level of suitability for a task. Several methods have been proposed to form cost metrics using background information on human and robot workers in addition to the specifications of manufacturing tasks (Johannsmeier and Haddadin, 2017; Nikolakis et al., 2018). To make use of this information, robust task planning methods are then required to generate subtask assignments and a task plan for the overall manufacturing task based on a manufacturer's constraints. These methods ensure that a worker is assigned tasks that they are capable of completing and are best suited to. Such task planners take differing approaches to the variables included in the objective function with some using only fatigue level of the human worker, such as in (Li et al., 2019), whereas others consider multiple variables including workload and operating costs, such as in (Johannsmeier and Haddadin, 2017; Nikolakis et al., 2018). In previous research, these task planners often are used within hierarchical systems with task planning occurring at a high level of abstraction by assigning subtasks to workers and lower level systems implementing robots in the task by carrying out the motion planning required.

Utilising robots as peers to human workers eases the strain on human workers by reducing their workload or ensuring they are not assigned tasks which could potentially cause them harm over long periods of time instead of merely providing assistance. This method can also be seen as a stepping stone to higher levels of automation, as tasks that are suited to automation can be completed by a robot worker resulting in partial automation of the manufacturing task. This also ensures that highly capable robots such as the Universal Robots UR5 can make a significant contribution to the manufacturing task and potentially increase the efficiency of the HR team completing the task. The disadvantage of this type of collaboration is that it can leave both human and robot workers idle whilst they are waiting for the other worker to complete their tasks. In an extreme case, if a robot worker failed and could not complete their tasks this could be highly disruptive to the completion of the task and require the human worker to intervene. This would necessitate the human

stopping their work to reset the robot worker's task and return the robot worker to a safe position.

2.3.4. Direct Collaboration vs Collaborative Working

These two views of the role of robot workers in the future of manufacturing provide very different visions on the level of HR collaboration in manufacturing tasks. Previous research shows a great deal of variation with differences including the level of autonomy of robots, the composition of HR teams and the level of direct interaction. Research such as (Aaltonen, Salmi and Marstio, 2018; Yanco and Drury, 2004, 2002) has attempted to classify levels of HR interaction to allow researchers to define the level of HR collaboration to consider.

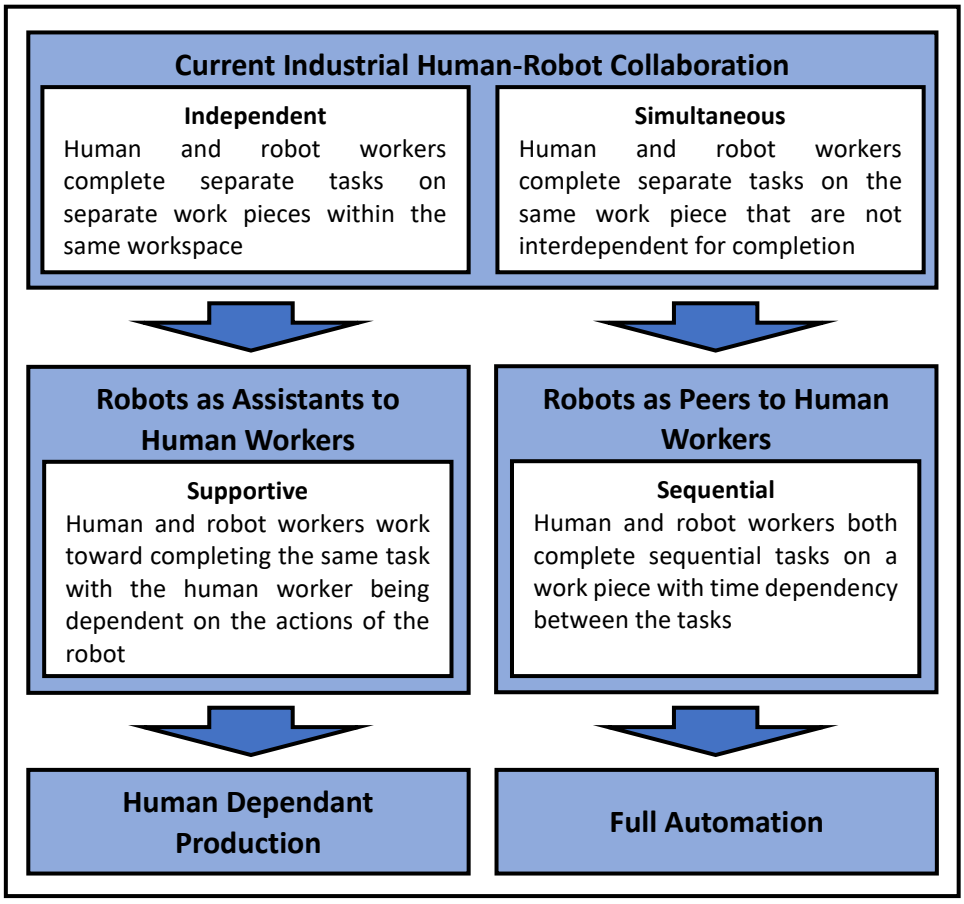

Figure 2.1: A diagrammatic overview of the potential paths for the implementation of HR teams in manufacturing tasks detailing the level of HR collaboration using the categorisation method proposed in (el Zaatari et al., 2019). This begins with current industrial implementations of HR teams, the two roles for robot workers proposed by current research and the type of production this would ultimately lead to.

Amongst the most recent work, (el Zaatari et al., 2019) developed a categorisation methodology to define the level of HR collaboration within a manufacturing workspace. Utilising this methodology, Figure 2.1 gives a diagrammatic overview of the potential paths for the implementation of HR teams in manufacturing and the categories of HR collaboration involved. As shown, humans and robots mostly coexist with each other in current industrial

HR collaboration with no interdependence between their tasks which was categorised as independent or simultaneous collaboration. This is limiting as the human and robot will complete the same tasks in rigid environments, with little adaptability to variation in human performance.

The use of robots as assistants to human workers was categorised as supportive collaboration. This role for robot workers leaves manufacturing tasks still very reliant on human workers for completion, as shown in Section 2.3.2. Implementing robots in this way would ultimately maintain human dependant production and be applied as a measure to prevent strain on a workforce. As a result, the utility of this role for robot workers is limited to the manufacture of larger and heavier products.

In contrast, the use of robots as peers to human workers was categorised as sequential collaboration. As discussed in Section 2.3.3, implementing robots in this way makes the completion of manufacturing tasks equally dependant on human and robot workers. This implementation of robots would ultimately provide a path to higher levels of automation as robot capabilities improve. This role also allows appropriate implementation of highly capable robots whilst enabling the reconfiguration of workers to optimise the efficiency of a manufacturing task. It was chosen to pursue this role for robot workers in this research due to these advantages to provide a novel adaptive way to optimise the implementation of HR teams in manufacturing tasks.

2.4. Task Planning in Robotics/Manufacturing

Before reviewing previous research into online and offline task planning for HR teams, it is necessary to conduct a broader review of task planning methods in manufacturing as these are often used as a part of such task planning methodologies. The methodologies presented in this section encompass ways to generate task sequences and/or task assignments in manufacturing processes, often considering optimality.

2.4.1. AND/OR Graph Search

A common method for representing assembly or disassembly tasks in manufacturing are AND/OR graphs which were first applied in (de Mello and Sanderson, 1986) for use with robot assembly cells. AND/OR graphs are directed hypergraphs containing all possible assembly or disassembly sequences for a product. In these graphs, nodes represent possible assemblies, only appearing once if made of the same component parts, and hyperarcs between them representing assembly or disassembly operations. In this representation, the root node represents the fully assembled product whilst terminal nodes represent its base components. Solution trees from the root node of the AND/OR graph represent possible assembly and disassembly sequences for the assembled product.

Traditional use of this representation in manufacturing provides weightings for hyperarcs representing operations proportionally according to a desired metric. This means the optimal assembly or disassembly sequence is the solution tree with the lowest combined weight (de Mello and Sanderson, 1986). Finding the optimal solution tree requires a search algorithm capable of traversing the AND/OR graph for which the A* search algorithm is commonly used (Knepper et al., 2014; del Valle and Camacho, 1996).

Although designed for assembly sequencing this methodology has also been used for the assignment of tasks in multi-robot teams such as in (del Valle and Camacho, 1996). Here hyperarc weightings for assembly operations are execution times calculated by using tool change time for a robot in addition to the required execution time for the resources to complete the operation. The optimisation metric was then considered as minimising the execution time for the assembly task and the A* search algorithm was used to do this. This methodology also allowed task planning in situations where sequential or parallel task execution was possible.

2.4.2. Petri Nets

Another method for modelling manufacturing tasks is Petri nets which were first applied to model assembly tasks in (Zhang, 1989). Petri net models of dynamic systems are composed of a net structure representing the static part of the system and a marking that represents a distributed overall state on the structure (Rosell, 2004). This net structure consists of a weighted directed bipartite graph containing a set of transitions and places connected by weighted arcs. Markings move around the net by firing transitions which allow them to move to new places. Transitions can only fire if all input places linked to the transition contain the number of markers specified by the weighting of the arc between them and the transition. The fired transition deposits tokens in each output place according to the weightings of the arc between it and the output places.

Petri nets are often used in assembly task planning since their structure natively enables the inclusion of preconditions for assembly subtasks via the defined transitions. Firing a transition represents a task being completed and a new place representing an assembly state being reached. Early methods such as in (Zhang, 1989) generated assembly plans by consecutively firing transitions to traverse a petri net with rules in place to avoid conflicts when transitions could be fired simultaneously. Later works proposed to translate AND/OR graphs, as described in Section 2.4.1, and directly (Suzuki et al., 1993) or indirectly (Cao and Sanderson, 1998) transform them into Petri nets. Works such as (Suzuki et al., 1993), proposed using linear programming with mathematical representations of Petri nets to find an optimal assembly sequence for subtasks in an overall assembly task. This allowed the use of more

powerful algorithms such as the simplex algorithm to optimise a linear objective function rather than the graphical search algorithms which must keep traversing an AND/OR graphs until an optimal assembly sequence is found.

Petri nets have evolved further into other models such as expert Petri nets (Zha and Lim, 2000) or timed Petri nets (Gu, Bahri and Cai, 2003; Lee and DiCesare, 1994) to further enhance their capabilities in planning applications. Expert Petri nets allow greater levels of task knowledge to be utilised by a Petri net whereas timed Petri nets include timings in the firing of transitions by adding a delay before the output is achieved to aid scheduling approaches.

2.4.3. Hierarchical Task Network Planning

Hierarchical Task Network (HTN) planning is a well-established abstract planning technique that has been applied over multiple domains (Ghallab, Nau and Traverso, 2004). Such techniques include an initial world state description, a task network that represents the target objective and domain knowledge consisting of networks of primitive and complex tasks (Georgievski and Aiello, 2015). The included task network is a hierarchy of tasks which is composed of executable primitive tasks and compound tasks that can be decomposed until they form a series of primitive tasks. The HTN planning process operates by decomposing the included initial task network until all complex tasks are decomposed into primitive tasks. This fully decomposed task network represents a solution to the task planning process as a series of primitive tasks that could be applied to the initial world state.

Numerous implementation methods for the HTN planning principle have been proposed with examples including SHOP2 (Nau et al., 2003) and UMCP (Erol, Hendler and Nau, 1994). HTN planning has also been used in robotics with implementations such as the SAHTN algorithm (Wolfe, Marthi and Russell, 2010) which combined task and motion planning for a robot or the HATP planner (de Silva, Lallement and Alami, 2015) which considered task planning for multi robot and agent teams.

2.4.4. Metaheuristic Based Optimisation Algorithms

Metaheuristic search algorithms allow efficient exploration of a search space to find near-optimal solutions to a problem (Blum and Roli, 2003) whilst not being problem specific. Some metaheuristic algorithms make use of memory to guide the search algorithm or use heuristics as domain specific knowledge to guide the search algorithm. This has led to the application of metaheuristic search algorithms in manufacturing or robotics problems. However, as with Hierarchical Task Networks, metaheuristics represent a search algorithm principle with multiple algorithms being developed. Multiple metaheuristic search algorithms have been proposed with common search algorithms applied in task planning including the Genetic algorithm, Ant Colony

Optimisation and Particle Swarm Optimisation. This methodology has often been applied in assembly line balancing problems with factors such as task assignment (Daoud et al., 2014; Dziki and Krenczyk, 2019) or the evaluation of assembly plans (Bonneville, Perrard and Henrioud, 1995). Metaheuristic search algorithms have often been shown to operate as local or global search algorithms. However, modern algorithms such as the Gravitational Search Algorithm (Rashedi, Nezamabadi-Pour and Saryazdi, 2009) instead widely explore the solution space in early iterations but exploit the best solutions through local searches of the solution space in later iterations.

2.4.5. Summary of Methods

The methodologies presented in this section showed various ways of modelling manufacturing processes as a means to optimise the process. Whilst methods such as searching AND/OR graphs or Petri nets represent well applied methodologies they are highly problem specific, requiring a new AND/OR graph or Petri net to be developed for each new assembly task. Despite this, these methods allow an exhaustive search of all potential task assignments or task sequences for the manufacturing task. Other methodologies such as metaheuristic optimisation algorithms or Hierarchical Task Networks allow task planning methods to be reapplied to other situations more easily. With metaheuristic algorithms, it was shown that these methodologies provide an efficient way to search a solution space, however, they do not guarantee that an optimal solution is found. Given the research aim to produce a generalised task planning methodology, metaheuristic search algorithms offer the best solution of these previous methodologies as they can be reapplied to new situations more easily than requiring a new model to be generated. Additionally, metaheuristic search algorithms can search large solution spaces more quickly and efficiently than traditional graph search methods that are executed by traversing a graph structure.

2.5. Offline Task Planning in Human-Robot Teams

One trend that has emerged in task planning methodologies for HR teams is offline task planning. Such methodologies plan entire tasks offline, before they are executed by a HR team, based on static indicators or models of worker performance. This allows optimisation of tasks as a whole but little adaptability to change. Across this section methodologies utilising offline task planning are explored, this includes methods for quantifying the performance of workers in addition to methods for generating task assignments and plans for HR teams.

2.5.1. Feasibility of Using Human or Robot Workers

To plan entire manufacturing tasks for a HR team, it is necessary to quantify the capabilities of human and robot workers to determine subtask assignments for workers to efficiently complete the overall assembly task. This often began with first determining if human and robot workers can feasibly complete each subtask. The result of this indicates if a specific worker, either worker or both workers together can be implemented to complete the subtask. The process of determining factors such as the workload of a task on a human worker is a well-established area of human factors research, methodologies for calculating this are shown in Chapter 18 of (Wilson and Sharples, 2015). Despite such well-established techniques, previous research outlined the need for a systematic approach to identify the utility of HR collaboration in manufacturing tasks.

A strategy for determining the feasibility and utility of workers to complete an assembly subtask was proposed in (Schröter et al., 2016) utilising capability indicators that determine the capability of a human worker compared to a robot worker. These capability indicators were generated for each criteria from a criteria catalogue, a weighting is then applied and the mean value of the capability indicators is taken as the overall capability indicator for a worker to complete an assembly subtask. This process allowed for comparison between human and robot workers in each subtask but was a manual procedure that would be used in the commissioning of a production process.

Other research attempted to develop similar strategies to determine the feasibility of humans and robots to complete manufacturing tasks via automated processes. A similar strategy proposed in (Bruno and Antonelli, 2018) considered the weight and displacement of an assembled part in addition to accuracy and dexterity requirements to complete the subtask. Each of these four subtask features were associated with a binary indicator which was manually given a value of 1 if the factor was relevant to subtask completion and 0 if not. These indicators were then used with a trained classifier to determine if a subtask could only be completed by a human, a robot, by either of them or with both of them collaboratively. This classification method did not consider quantifying the capabilities of workers in subtasks that could be feasibly completed by either a human or robot. However, a corresponding task assignment process was described to utilise their classification method. This first assigned tasks according to worker feasibility defined by the classifier then assigned tasks either worker could complete based on worker availability and task constraints.

2.5.2. Cost Functions to Evaluate Worker Capabilities

Section 2.5.1 showed that methodologies developed for determining worker feasibility to complete manufacturing subtasks often did not consider how to autonomously assign subtasks that either humans or robots could feasibly complete. Other research proposed using cost functions in such situations that consider multiple metrics to quantify worker suitability to complete manufacturing subtasks allowing an optimal set of task assignments to be determined. Significant previous work by Johannsmeier and Haddadin in (Johannsmeier and Haddadin, 2017) proposed using metrics such as subtask execution times, operational costs and risk factors to human workers. In addition, the concept of a worker profile was proposed that could map metrics such as attention level, general experience level and reliability to a cost function for a human worker. Whilst a worker profile would allow task assignments that are highly informed by worker capabilities, the metrics proposed are highly subjective and would be difficult to evaluate fairly. Additionally, factors such as reliability and attention level are likely to vary significantly over time and utilising a static variable would not appropriately illustrate such capabilities. In (Johannsmeier and Haddadin, 2017), separate possible cost functions are suggested for human and robot workers, containing execution time and power consumption metrics for a robot worker but instead containing anticipated attention level and workload for a human worker. This would be inappropriate as worker costs would reflect totally different capabilities and could be considered an unfair comparison of workers.

Similar quantification of capabilities are considered in (Lamon et al., 2019) which utilised metrics such as task complexity, agent dexterity and agent effort to determine worker costs to complete subtasks of an overall manufacturing task. In this case and in (Johannsmeier and Haddadin, 2017), worker feasibility to complete subtasks is not considered prior to cost evaluations. Instead, a worker is given an infinite cost if they are incapable of completing the subtask to it is not assigned to them. In other research such as (Nikolakis et al., 2018) manufacturing tasks are considered where it was predetermined that human or robot workers could feasibly complete all subtasks. In this case, a set of task assignments were assessed for optimality using factors such as the cycle time for the task, total completion time of tasks assigned to human workers, the total weight lifted by human workers across the task and the operating cost of the human and robot workers.

2.5.3. Modelling of Human Workers

The methodologies presented in Sections 2.5.1 and 2.5.2 often quantified the capabilities of human and robot workers using static variables. The use of static variables to quantify the capabilities of a robot worker is reasonable as it is well acknowledged in previous research that little variation

exists in their capabilities during a work shift, with the exception of a major hardware or software failure. However, this is not appropriate for human workers since their performance is highly variable across a work shift which can affect the efficiency of the HR team as a whole. Additionally, proposed cost function variables in (Johannsmeier and Haddadin, 2017) such as general experience level can be highly subjective and may lead to an unfair assessment of a worker's capabilities.

Previous research has attempted to integrate models of factors such as fatigue of human workers into task planning for HR teams as shown in (Hu and Chen, 2017; Li et al., 2019). Such methodologies allow the quantification of human workers capabilities to reflect the variable nature of their performance across a work shift. This was shown in (Li et al., 2019) as part of task planning for a HR team to complete batch disassembly of a gear pump for remanufacturing. Here subtasks that either a human or robot worker could feasibly complete were assigned based on the fatigue level of the human worker generated by models proposed in (Glock et al., 2019). Additionally, optimal task sequencing was determined utilising human completion times when subject to fatigue generated by a model proposed in (Digiesi et al., 2009).

Although not implemented in such methods, many other models have been proposed to determine the fatigue level of a human worker focusing on factors such as muscular or mental fatigue (Dawson et al., 2011). Such methodologies cannot fully solve the issue of representing the varying capabilities of a human worker as models cannot take into account unexpected events in a human workers life that could significantly affect their capabilities. Factors such as this or model inaccuracies would result in inaccurate quantification of a human worker's capabilities and may limit the effectiveness of a task planner. As a result of such limitations, some models have proposed using machine vision systems to detect visual cues as evidence of fatigue to better understand a human's fatigue levels (Ji, Lan and Looney, 2006).

2.5.4. Task Planning at High Levels of Abstraction

The metrics for quantifying human and robot workers capabilities outlined in Sections 2.5.1 to 2.5.3 are often utilised by task planners to determine task assignments and/or task sequences for manufacturing tasks completed by HR teams. Such task planners often operate at higher levels of abstraction and allocate entire subtasks to a human worker, a robot worker or a HR team to complete collaboratively. Lower level systems then allow robot workers to execute a task by implementing trajectory planning and other required functions to allow the robot worker to complete the task. Some lower level systems also include limited abilities to react to dynamic events such as a collision between a human and robot (Johannsmeier and Haddadin, 2017).

Task planning at this higher level of abstraction has been both manually defined and autonomously executed in previous research. Manual generation of task assignments was previously shown in (Schröter et al., 2016) and (Ranz, Hummel and Sihn, 2017). However, this occurred during the process of determining the feasibility and utility of workers to complete assembly subtasks discussed in Section 2.5.1. In both cases task assignments were rigid as a result and could not change to reflect changing worker capabilities or errors in production without again being manually redefined. In comparison in (Bruno and Antonelli, 2018), task indicators were generated manually when determining workers feasibility to complete assembly subtasks, however, the rest of the task assignment process was done autonomously. Optimisation was not considered in when both human and robot workers were suitable to complete the subtask and task assignments were based on worker availability whilst respecting task precedence relations for subtasks.

Optimality of task assignments was considered in (Johannsmeier and Haddadin, 2017) with the autonomous generation of task assignments. Here an assembly task was represented by an AND/OR graph and the A* graph search algorithm was utilised with costs for a worker to complete each assembly subtask to determine an optimal set of task assignments for the HR team. A similar approach was posed in (Lamon et al., 2019) but this approach utilised different cost functions to determine worker capabilities as described in Section 2.5.2. Such methods of representing tasks and planning their execution are also commonly used in wider research in the field of manufacturing engineering with teams consisting entirely of robots or other manufacturing processes as shown in Section 2.4.1. The novelty of the research presented in (Johannsmeier and Haddadin, 2017; Lamon et al., 2019) compared to previous research is that task planning is applied to HR teams with quantification of human and robot workers capabilities. Although the methods presented allow capability based task planning in HR teams, they do not consider reassigning manufacturing subtasks throughout a work shift. This can prove disadvantageous for HR collaboration as it means task assignments are not altered to reflect changes in a human worker's capabilities as with the quantification of worker capabilities in Section 2.5.2.

The need for task planning to reflect the variable nature of human capabilities has been identified in previous research (Casalino et al., 2019a; Ranz, Hummel and Sihn, 2017) and this has been partially achieved by using models to simulate a human worker's performance across a work shift. In (Li et al., 2019), a human worker's fatigue level was modelled to determine task assignments and sequencing as their performance declines across a batch of disassembly tasks. As described in Section 2.5.3, this methodology assigned tasks to workers based on capabilities by assigning tasks that either humans or robot could complete based on the fatigue level of the human worker. For each generated set of task assignments, the optimal disassembly sequence was

determined using the Bees algorithm, a metaheuristic search algorithm, with an objective function that optimised the total completion time of the task. This was possible by generating expected human completion times via a model for human completion times in repetitive tasks under the effects of fatigue proposed in (Digiesi et al., 2009). Utilising a metaheuristic search algorithm ensured a quick execution time and good solution is found to the task sequencing problem, however, this does not guarantee an optimal solution is found.

In comparison, (Hu and Chen, 2017) proposed to use a stochastic Petri net framework to model the impact of fatigue on the dynamics of manufacturing processes with stochastic time and event driven dynamics. This was transformed into a continuous-time Markov decision process and using a cascaded composition of a human fatigue model and the manufacturing process model allowed the task planning problem to be solved through solving a linear program. The disadvantages of the methodologies proposed in (Hu and Chen, 2017; Li et al., 2019) is the heavy dependence on the fatigue models used means that if the model is inaccurate the task plans generated may not obtain the best performance for the HR team.

2.5.5. Summary of Methods and Definition of State of the Art

In this section methodologies have been presented for planning HR collaborative completion of manufacturing tasks offline over various levels of abstraction. Here the advantages of hierarchical approaches were shown by allowing task assignment and planning at higher levels of abstraction based on worker capabilities without considering factors such as replanning robot trajectories. This can be advantageous over methods that integrated robot motion planning into the task assignment process as it enables the possibility of easily reapplying the task planner to various manufacturing tasks. Two different areas of research should be considered as state of the art for different aspects of the task planning problem.

Methods such as those proposed in (Johannsmeier and Haddadin, 2017) should be considered as state of the art in terms of quantifying a worker's capabilities to inform task assignments. This conclusion can be drawn as they allow the assignment of tasks to an optimal worker based on multiple factors including worker performance, operational cost and workload. This means that task plans can be optimised to consider factors such as the effects of workload on human workers in addition to more traditional factors such as production rates. Given the research aim to optimise the implementation of Human and robot workers, multivariable cost functions offer a suitable solution to quantify worker capabilities to ensure this is achieved. The disadvantage of current methods is that cost functions are not updated as a human worker's

capabilities change across a work shift which must be addressed for these to be utilised.

It was instead shown that the state of the art for offline planning of manufacturing tasks are the methods proposed in (Li et al., 2019) which generate multiple sets of task assignments and sequences to be enacted across a work shift that factor in a human worker's fatigue levels. This showed the benefits of replanning a task across a work shift as it resulted in a lower fatigue level for the human worker and a lower total completion time for a set number of iterations of a disassembly task. Given the research aim to optimise the implementation of HR teams whilst allowing adaptability, such an approach is necessary to adapt the implementation of the HR team across a work shift as the capabilities of workers change. Despite these capabilities, the task planning method proposed in (Li et al., 2019) assigned tasks that could be completed by a human or robot worker based on only the fatigue level of the human worker. This is limiting as there are many other factors that could determine the optimal worker for such a task which may result in a different assignment. Similarly, the objective function for task sequencing was based solely on the total execution time for a task iteration which is again limiting as many other factors may be important to manufacturers. This reinforces the importance of the methodology proposed in this research to use multivariable cost functions to assess worker capabilities to ensure tasks are assigned to the optimal worker.

Both the state-of-the-art methodologies presented in this section face the limitation that they infer or model the performance of human workers. This means that the optimum task plans generated will not be suitable for the HR team if a worker has an unexpected change in performance or capabilities across a work shift. To address these limitations, methods must be developed in this research to utilise online data to similarly quantify the capabilities of workers.

2.6. Online Task Planning in Human-Robot Teams

The other emerging trend in task planning methodologies for HR teams is online task planning. Online task planning methodologies operate by assigning tasks one by one as they are being executed, allowing adaptation to change in performance or composition of HR teams but not necessarily optimisation of tasks overall. Across this section two types of online task planning are explored. This includes planning by observing the world state of a HR team's collaborative workspace in addition to semi-online task planning based on updated information on the HR team.

2.6.1. Planning by Observing the World State

A common method for implementing online task planning operates by assigning tasks on a task by task basis by observing the world state of the HR team's workspace utilising sensors such as depth cameras. As with the implementation of robot assistants described in Section 2.3.2, such methods adapt a robot's actions around those of the human worker allowing the human worker to complete tasks as they wish.

An example of this task planning theory was shown in (Riedelbauch and Henrich, 2017) where observation of the world state of the work environment was used to coordinate flexible HR teams. This utilised preconditions and post conditions linked to operations to determine if they were available for completion or had been completed. In hand mounted sensors allowed a robot worker to use this methodology to observe the world state and determine an available operation for it to execute to contribute to the manufacturing task. This methodology was further improved in (Riedelbauch and Henrich, 2019) to make the world model human aware by determining a measure of certainty that an object is still in its stored location since the robot last observed the world space, based on human motion and previously detected task progress. This was combined with a novel action selection algorithm to execute operations that were likely to succeed and trigger perception actions to refresh the world model. This proposed methodology allowed flexible HR teams, enabling a human worker to leave the workspace during the execution of a manufacturing task whilst the robot completes any remaining operations which would not be possible with offline planning methods. Despite this, the methodology had the disadvantage that it did not consider optimality of task assignments so they may not represent the optimum implementation of human or robot workers in a collaborative task. Additionally, such collaboration could be considered highly unstructured leading to different operations being completed by the human and robot worker in each task iteration.

Other methodologies did seek to include optimality such as in (Wang et al., 2018) where a sequential collaborative assembly was considered between a human and a robot worker. Here through observation of the world state, a cost function gave a cost for an operation performed by a human worker which was then used to plan the robot worker's action to ensure the total collaborative assembly cost was minimised. Although this methodology considered the optimisation of robot actions in online task planning, as with (Riedelbauch and Henrich, 2017, 2019) it did not consider control of the implementation of the human worker in HR collaborative tasks. This posed the disadvantage that the human worker may not be implemented in the task in an optimal way and as a result although optimised in this context, the robot's implementation may not be optimal either.

2.6.2. Planning Given Updated Worker Information (Semi-Online Planning)

In other research, task planning methodologies were shown for HR teams that consider offline task planning but utilising online data regarding the current world state of the manufacturing environment or the capabilities of human workers. Such methodologies consider various levels of implementation and often were a part of a larger hierarchical task assignment structure such as those seen in Section 2.5.4. Although discussed amongst online task planning methodologies, this type of task planning is referred to as semi-online task planning in this thesis due to the way it bridges the gap between fully offline and fully online methodologies. As such this methodology can be considered as analogous to the concept of HR collaboration in itself by combining the benefits of online task planning, by reacting to changes in worker capabilities, and offline task planning, considering the optimisation of tasks as a whole. Such task planning approaches for HR teams can be considered as state-of-the-art methods, however, research in this area is still quite limited with the majority of research focusing on the fully offline or online methods present in Sections 2.4 and 2.5.1, respectively.

A very basic implementation of such a methodology was proposed in (Nikolakis et al., 2018) where subtask assignments for an assembly task could be updated given the current availability of workers. This was enabled through the use of a monitoring system that could determine worker availability and trigger replanning of the task based on new knowledge of the composition of the HR team. To illustrate this, an example was given for the assembly of a turbocharger where a human worker was determined to be absent during the completion of the first subtask of the assembly. The monitoring system triggered a need for the remaining task assignments to be re-evaluated in response, allowing any subtasks assigned to the missing worker in the initial set of task assignments to be redistributed amongst the remaining workers based on their capabilities. This method had the advantage that task assignments can be updated given a change in the HR team but unlike the methodologies presented in Section 2.6.1 considered the optimal use of all available workers. Despite these capabilities, the implementation of the methodology proposed by (Nikolakis et al., 2018) did not consider the changing capabilities of workers across the work shift and still used static variables to quantify a human worker's capabilities.

A different level of implementation of this methodology was seen in (Casalino et al., 2019b) where subtask scheduling was updated based on the completion times of human and robot workers. This was enabled by using RGB-D sensors such as the Microsoft Kinect to detect when the human worker enters or exits areas of a workspace to determine their completion time for subtasks. These completion times were used to update a model to predict the

completion time for a human worker to complete their individual assigned subtasks of the overall manufacturing task. This data was used within a time Petri net that modelled the HR team's completion of the task from which an optimal task sequencing was generated using Monte Carlo methods. This implementation of semi-online task planning had the advantage of adapting subtask sequencing to a human worker's current performance whilst optimising the sequencing as a whole to improve the production rate of the HR team.

Although this implementation of semi-online task planning demonstrates a promising approach to task planning utilising on current worker performance data, it poses several limitations which would limit its applications. One such limitation was that the system does not consider the reassignment of subtasks with variation of worker performance which limits the capability of the task planner to react to declining performance of a human worker. This was reinforced by completion times being the only form of online data collected, meaning discrete events such as errors could not be assessed to determine a worker's capabilities. Finally, this implementation of the methodology used time Petri nets to model the HR team's completion of the manufacturing task. This can be considered as a disadvantage since this would require an expert to develop a new time Petri net to model the HR team if it were implemented on a different manufacturing task.

2.6.3. Summary of Methods

Across this literature review, several methodologies have been discussed for online task planning in HR teams given current information about the completion of tasks. The first of these task planning methodologies described in Section 2.6.1, allowed a robot worker to independently determine and execute subtasks available for completion in a manufacturing task completed by a HR team. This had the advantage that HR teams could be highly dynamic to the extent that the composition of the team could change during an iteration of a manufacturing task. Despite the flexibility offered, this methodology should not be considered for efficient implementation of HR teams as tasks are assigned to robot workers on a task by task basis. Although optimality was considered in some research for such task assignments, this could still lead to non-optimal implementation of the human and robot workers in addition to the task becoming unstructured.

The semi-online task planning methods in Section 2.6.2 instead showed a more promising methodology for efficiently implementing HR teams in manufacturing tasks. This was achieved by optimising manufacturing tasks as a whole whilst considering individual workers current capabilities using data gathered from the shared workspace. The current research in this field is still quite limited and focuses on the use of specific data to achieve a very specific

task planning goal. This was shown through assigning tasks based on worker availability in addition to static measures of their capabilities (Nikolakis et al., 2018) or task sequencing based on the current performance of human and robot workers (Casalino et al., 2019b).

In this research, the use of semi-online task planning will be further explored as its operating principle provides a way to fulfil the research aim of optimising the implementation of human and robot workers whilst allowing adaptability. However, its limitations must first be addressed to improve its capabilities. To increase the utility of semi-online task planning, it is necessary to implement multivariable cost functions, as seen in Section 2.5.2, utilising multiple sources of online production data to quantify the current capabilities of human and robot workers. Task planning must then occur throughout the work shift considering optimisation of task assignments in addition to task sequencing.

2.7. Knowledge Gaps

From the literature review presented in this chapter, it was determined that a knowledge gap existed for dynamically planning tasks for HR teams based on the variability in performance and capabilities of workers across a work shift. Here, it was determined that semi-online methods of task planning, such as those described in Section 2.6.2, had the potential to allow efficient and adaptable implementation of a HR team in a manufacturing task. Such approaches have so far focused on task replanning based on specific factors influencing HR teams and the effect of these on the capabilities and performance of the HR team. This was shown in previous research that has focused on the assignment of tasks based on available workers or the sequencing of tasks based on their current completion times.

It was shown that existing semi-online task planning methodologies did not consider using multi-variable cost functions, such as those for offline task planning shown in Section 2.5.2, which update a cost for a worker to complete a manufacturing subtask using data from the HR workspace. In addition to this, existing semi-online task planning methodologies do not fully consider the reassignment of assembly subtasks with the changing capabilities of human workers across a work shift. This was considered for offline tasks as seen in (Li et al., 2019). However, the replanning lacked structure since it was executed for every iteration of the disassembly task considered.

Given these factors, the knowledge gap can be further refined as the requirement of a generalised methodology to allow for semi-online structured task replanning of a manufacturing task over a work shift for a HR team given the variability of workers capabilities. Such a methodology must utilise multi-variable cost functions that can quantify a worker's performance and capabilities and updating this assessment using online data from workers

completing the task. This must then be used by an offline task planner to optimise the assignment and sequencing of subtasks of an overall manufacturing task.

2.8. Chapter Summary

This chapter began with a description of the current uses of automation in industry to illustrate the importance of the research outlined over the literature review. Following this, a detailed discussion was presented outlining two proposed roles of robots in a HR team as either an assistant to a human worker or working as their peer. Given the choice of investigating methodologies for utilising robots as peers to human workers in this research, a description of existing commonly used task planning methodologies was presented for robots in manufacturing. The state-of-the-art research in offline and online task planning was presented for use when robot workers act as peers to human workers in HR teams. Utilising this discussion of state-of-the-art methodologies, a knowledge gap was defined for semi-online task planning to allow the efficient and adaptable implementation of HR teams explored within this thesis. This knowledge gap was further refined as the requirement for a generalised methodology for structured task replanning of a manufacturing task for HR teams over a work shift given the variability of workers capabilities. These challenges are addressed in this thesis through the definition of a system architecture for a generalised task planning system in Chapter 3 that can be applied to various manufacturing tasks and configurations of HR teams. This is followed in later chapters by the development of the core methodologies enabling the task planning system and verification of the capabilities of intelligent semi-online task planning across a work shift for a HR team.

3. Research Methodology

3.1. Introduction

Building on the knowledge gaps identified in the literature review in Chapter 2, this research is motivated by improving the implementation of Human-Robot (HR) teams in manufacturing tasks. In this chapter, the research methodology utilised in this thesis is described, beginning with details of the structure of research. Section 3.2 first defines the objectives of this research in addition to an outline of the research approach and how this defines the structure of the thesis. Given the research approach, Section 3.3 then describes the selection of research methods providing a reasoning for the choice of approaches or technologies used in this research. Following this, Section 3.4 describes how manufacturing tasks are defined and the level of decomposition required for task planning with a HR team. Next, in Section 3.5, a system architecture is proposed for the generalised methodology of semi-online task planning. This is required in order to define how such an approach would be implemented in a real-world situation, and where the core enabling methodologies developed in the thesis would fit in to such a system. Finally, a brief overview is given in Section 3.6 of how the proposed dynamic cost functions operate in addition to their structures and the weighting of variables.

3.2. Structure of Research

3.2.1. Research Aims and Definition of Objectives

This work aims to provide a methodology to implement robot workers amongst human workers as part of a HR team, whilst optimising implementation of both workers and allowing adaptability. Utilising the knowledge gaps defined in Section 2.7 in addition to the aims of this research, it is possible to define the objectives of this research. First, based on the limitations of utilising online production data to quantify the current capabilities of a worker, the following research objectives are defined:

- Formulate dynamic cost functions for human and robot workers, consisting of variables that can use continuous or discrete production data to quantify the capability of each worker to complete each subtask of an overall manufacturing task;
- Develop mechanisms to update the output cost for a cost function variable given online data obtained from the collaborative workspace over iterations of a manufacturing task. These mechanisms must ensure the output cost from a cost function variable accurately quantifies the capabilities of human and robot workers.

Next, given the limitations of task planning in using such online production data, the following objectives are defined:

- Produce a task planning methodology to find an optimum set of task assignments and task plan for a HR team given the costs generated for each subtask whilst respecting task constraints and minimising worker idle times;
- Implement mechanisms to ensure that subtasks are assigned to optimal workers if there is a significant difference in worker capabilities;
- Implement mechanisms to trigger task replanning at appropriate intervals but only if worker costs changes indicate this is necessary.

3.2.2. Outline of Research Approach and Thesis

Given the development of the research focus and objectives over this section, it is necessary to define an overall research approach to achieve the research objectives in Section 3.2.1. The overall research approach of the work presented in this thesis is graphically outlined in Figure 3.1. This shows that the research approach began with the thorough literature review presented in Chapter 2, which resulted in the decision to explore HR teams, where robots are utilised as peers to human workers. The knowledge gap existing for task planning, using a semi-online methodology in such teams was then identified.

In the next phase of the research given in this chapter, the research objectives are defined utilising the identified knowledge gap. From this it is also necessary to define manufacturing tasks considered for task planning in addition to the proposed system architecture for the semi-online task planning system. As a part of this definition, it is necessary to define the elements of this system developed within this research, in order to validate the proposed semi-online task planning methodology for HR teams. Finally, it is also necessary to define the structure of the cost functions used within this task planning system in addition to their operating principle.

The final phase of the research focuses on the development of the core elements of this methodology required for it to function correctly, which are explored and tested in Chapters 4, 5, 6 and 7. This includes the development of cost function variables for continuous data in Chapter 4 and for discrete data in Chapter 5. These are then implemented into a task planning methodology which is tested for accuracy and execution time in Chapter 6. Finally, in Chapter 7 the task planning methodology was tested to determine the capabilities of such a methodology to improve the efficiency of a HR team consisting of a single human and a single robot worker in an example manufacturing task.

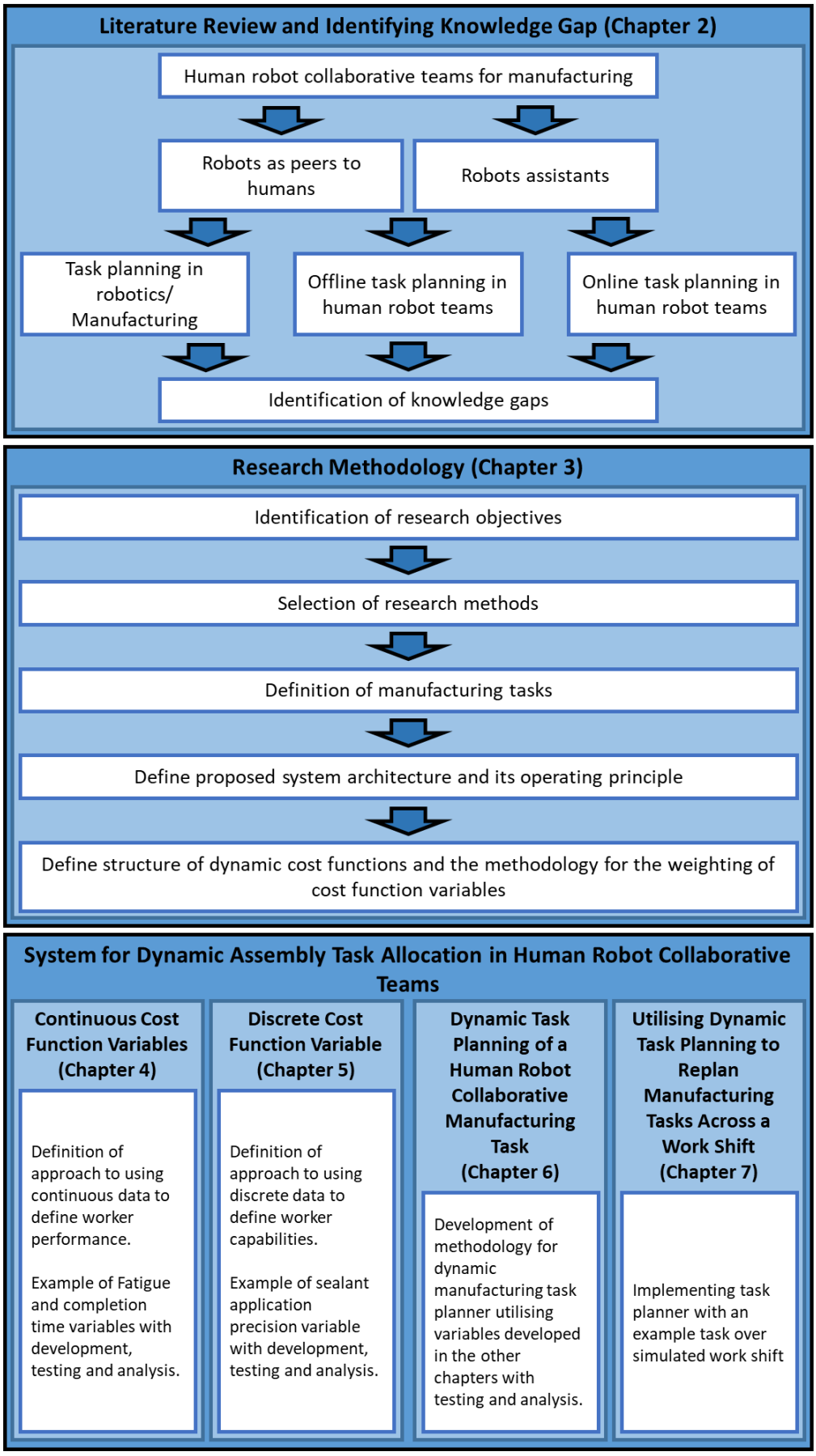

Figure 3.1: Outline of the research approach.

3.3. Selection of Research Methods

To fulfil the research objectives in Section 3.2.1 and satisfy the research aims, it is necessary to select the research methods and technologies that will be used to achieve this. First, it is necessary to select a methodology that can accurately quantify the capabilities of workers utilising online production data. As described in the research objectives, multivariable cost functions that utilise online production data are chosen to achieve this. These will be used as they allow costs to be generated for workers whilst considering multiple factors relating to their performance and capabilities, as highlighted in Section 2.5.2 of the literature review. As stated in the research objectives, the variables used by such cost functions should be able to utilise continuous and discrete online production data. Here continuous data will be used by continuous variables to assess gradual changes in worker performance over a work shift, with completion times in subtasks chosen to achieve this. Completion times are chosen as the completion times of human workers should gradually change over a work shift with their performance levels, allowing inference of their current performance level. In contrast discrete data will be used by a discrete variable to assess sudden significant changes in a worker's capabilities utilising data from individual iterations of an example sealant application subtask. Such an example task is chosen as sudden significant errors in individual iterations of a subtask could imply a change in a worker's capabilities.

Second, it is necessary to select a methodology to obtain an optimum set of task assignments and task plan based on costs generated for workers. Section 2.5 of the literature review highlighted that metaheuristic search algorithms offered a promising to efficiently and intelligently search a solution space. As shown in Section 2.4.4 of the literature review many metaheuristic search algorithms exist such as Ant Colony Optimisation, Particle Swarm Optimisation and the Genetic Algorithm. All of these algorithms intelligently search solution spaces through an iterative process by exploring the solution space based on knowledge of the solution space. However, in this research the Discrete Gravitational Search Algorithm (DGSA) (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014) will be used to form a generalised task planning methodology. This algorithm is chosen as it prioritises exploration of the solution space in early iterations while exploiting the best solutions found in later iterations through prioritising local searches. Additionally, this algorithm has previously been applied to the Travelling Salesman Problem, a graph search problem that shares many similarities to the task planning problem. The task planning problem can be considered as a comparable graph search problem by considering each subtask as a node with task precedence constraints defining the vertices between them. This allows an optimal task plan to be found by searching for the optimal Hamiltonian Path between the first subtask and last subtask where all other subtasks are visited exactly once. This method is also

chosen since it can be used for permutation problems such as finding the optimal set of worker task assignments.

3.4. Definition of Manufacturing Tasks

It is also necessary to determine how manufacturing tasks are defined in this research and the level of task decomposition necessary to plan a manufacturing task for a HR team. First, it is possible to define the level of task decomposition based on the representations used in previous research such as that seen in (Nikolakis et al., 2018), by focusing on task planning in HR teams. In this work, a manufacturing task is broken down into a series of subtasks consisting of one or more primitive actions. Here primitive actions consist of an interaction that alters the state of a component, such as tightening a bolt, moving a component or checking the specifications for a subtask. Subtasks are instead considered as actions that move the product from one partially assembled state to the next. A transformation to the next partially assembled state can be achieved by adding components to the partially assembled product or a subassembly of it through pick and place operations. Additional transformations could include fastening components in place using bolt tightening subtasks or applying sealant to the partially assembled product. Decomposing manufacturing tasks to this level ensures that subtasks are assigned at a level where they have a significant effect on the manufacturing task ensuring a worker making a meaningful contribution to its overall completion.

Although subtasks are assigned at this level, it is also necessary to break down some subtasks further into their primitive elements. This is to allow the weighting of cost function variables used to define the cost for a worker to perform a subtask of the overall manufacturing task. In particular, this approach would be necessary for subtasks, such as a pick-and-place operation, which require multiple primitive elements such as checking the task specification, locating, retrieving parts, and then placing the parts in the correct position. This is not required for subtasks, such as tightening a bolt, which would also be considered as a primitive task.

3.5. Proposed Architecture Overview

3.5.1. Operating Principle

It is first necessary to define the operating principle of the proposed task planning system to allow the definition of its structure. It was determined through the literature review in Chapter 2 that a task planning methodology is required that utilises online data to quantify the capabilities of workers whilst still planning manufacturing tasks as a whole. To achieve this, it is proposed here that the task planning system should be formed as presented in the

system diagram in Figure 3.2. Utilising such a methodology, it is proposed that tasks are firstly planned offline, using historic data to generate costs for each worker to complete each of the manufacturing subtasks. The HR team then collaboratively complete the manufacturing task by using the initially generated set of task assignments and task plan. As this occurs, data must be collected on their performance to allow the costs for workers to complete subtasks to be updated. After a set number of complete task iterations, the task must again be planned offline to find the optimal set of task assignments and task plan given the current worker costs. This process will continue until a human worker enters a rest period, where the robot worker should take over all tasks, or until the work shift is complete.

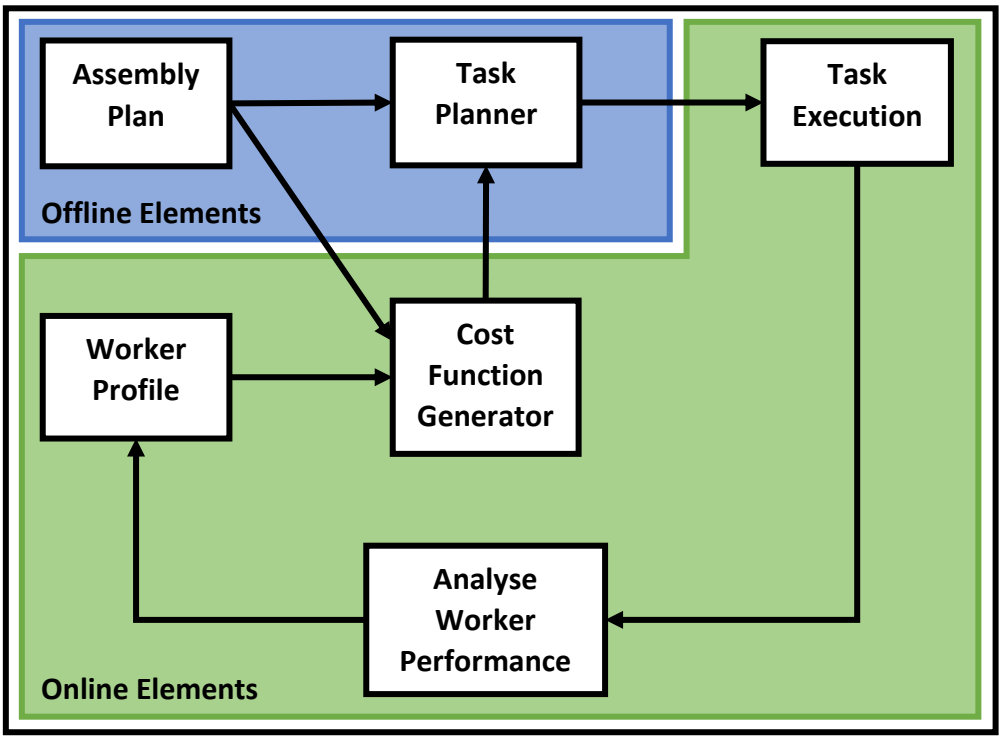

Figure 3.2: A diagrammatic simplified overview of the task planning system where the boxes represent elements of the system and the arrows represent the flow of data. Additionally, this diagram highlights the elements of the system that operate online and the elements that operate offline.

3.5.2. Inputs to the System Architecture

The input to the proposed task planning system is given by an abstract assembly plan in addition to a worker profile as shown in Figure 3.2. Here an assembly plan should contain all necessary information to plan a manufacturing task and allow the workers to execute it. This assembly plan should include an abstract representation of the manufacturing task that is understandable by a human and useful for robot workers but can easily be defined and utilised within a search algorithm as defined in Chapter 6. In addition to this, other necessary information should include data required to calculate cost function variables for each subtask, with examples given for continuous variables in Chapter 4, and discrete variables in Chapter 5. Finally,

the assembly plan should contain the instructions on how each subtask should be completed. However, the definition of a formalism for this and the overall assembly plan file structure is considered outside of the scope of this research.

The other input to the system, the worker profile, should be considered as a profile that stores the data collected from the shared workspace from the human or robot worker completing assembly subtasks. In this research, the data that would be contained in such a worker profile is simulated, with examples given in Chapters 4 to 7. However, defining a formal structure for a worker profile is also considered outside the scope of this research.

3.5.3. System Processing of Data

The processing elements of the task planning system are formed by a cost function generator and a task planner which form the core focus of this research and are utilised to demonstrate the principles of semi-online task planning. The cost function generator will take input information from the assembly plan and worker profiles to generate a cost for each worker to complete each of the subtasks of an overall manufacturing task. In this research, continuous and discrete variables are defined in Chapters 4 and 5, respectively, whilst the methodology for defining the weighting of these variables in a cost function for a subtask is given in Section 4.2. The costs defined by the cost function generator for each worker to complete each subtask are then used as input for the task planning processing element which also retrieves information on the manufacturing task from the assembly plan.

The task planner used by the system contains the search algorithms necessary to determine the optimal set of task assignments and task plan for the HR team given the input information. In comparison to some of the task planning methodologies presented in the literature review in Chapter 2, the task planner must be adaptable to be used with various manufacturing tasks and workers.

3.5.4. Task Execution by the Human-Robot Team

Given a generated set of task assignments and task plan for the HR team, it is next necessary for them to be transferred to a system that can manage the execution of the task by the team. This requires several complex elements to successfully orchestrate, which can form a complex research problem in itself and is considered beyond the scope of this research which has the main focus of the task planner. Whilst the task is being executed, it is also necessary to capture information from the collaborative workspace. This again can form a complex research task in itself if done autonomously for human workers, but possible methodologies are presented for the capture of discrete data in Chapter 5. It is assumed, with continuous variables, that it is possible to capture data such as completion times from human workers by having them

press a button when they start and finish a task instead of using more complex vision-based methods.

3.6. Dynamic Cost Functions

In this research, dynamic cost functions will be utilised to quantify the performance and capabilities of human or robot workers as they vary across a work shift as defined in the research objectives in Section 3.2. To achieve this, continuous and discrete cost function variables are needed to interpret continuous and discrete changes, respectively, in online production data. These variables must interpret the data separately as continuous variables must use gradual changes in data that occur slowly across a work shift, whereas discrete variables must detect instantaneous changes in data that occur during a single task iteration. As stated earlier in this chapter, examples of continuous variables will be developed in Chapter 4 and an example of a discrete variable will be developed in Chapter 5. Before defining the proposed cost function variables, it is necessary to define the overall structure of the dynamic cost functions used in the cost function generator.

Cost functions must be adaptable to various production requirements in addition to various human or robot workers, so it is necessary to use a basic generic format for a cost function. To achieve this, the cost function to generate a cost, c_j, for a worker to complete a subtask *j* of an overall manufacturing task is defined as

$$c_j = \sum_{i=1}^{k} g_{i,j} f_{i,j} \tag{3.1}$$

where $f_{i,j}$ represents the i^{th} cost function variable and $g_{i,j}$ defines its weighting for *k* available cost function variables. This generalised definition allows the adaptability of the cost function generator to various manufacturing tasks and workers but, the weightings, $g_{i,j}$, used by the cost function must be equally adaptable.

To ensure the cost generated is relevant to the worker and subtask it is applied to, a methodology is proposed that allows weightings to be easily defined as a part of the initial implementation of the task planning system to a manufacturing task. As assignable subtasks can be broken down into primitive tasks, as stated in Section 3.3, it is proposed to associate commonly used types of primitive task with relevant cost function variables. These associations allow a method analogous to a keyword search to be employed to determine cost function weightings that can be automated. For each subtask, it is necessary to search the primitive tasks it is composed of in order to determine which cost function variables influence the outcome of each primitive task. This allows the total number of primitive tasks which a cost function variable influences to be

defined as $\mu_{i,j}$ for the i^{th} cost function variable in the j^{th} subtask of an overall manufacturing task. Using these values allows a subtask's weighting to be defined as

$$g_{i,j} = \frac{\mu_{i,j}}{\sum_{v=1}^{k} \mu_{v,j}} \tag{3.2}$$

where there are *k* available cost function variables, which ensures all weightings sum to one for each subtask and are also normalised between zero and one.

3.7. Chapter Summary

In this chapter, it was possible to exploit the knowledge gaps found in Chapter 2 in addition to the overall aims of this research, to define the research objectives that will be attempted to be met in the thesis. From these objectives, it was then possible to define the research approach to implement and how this reflected in the structure of the thesis. Following this, an architecture was proposed for a task planning system to enable the application of semi-online task planning to a HR team collaboratively completing a manufacturing task across a work shift. Although the focus of this research is not to develop such an architecture in a readily implementable form for industry, this chapter highlights where the core methodologies developed in this research should be integrated into a semi-online task planning architecture. In addition, the structure of the dynamic cost functions utilised in this research is introduced, by including a methodology of weighting cost function variables to ensure cost functions can be adapted to various workers and manufacturing subtasks.

4. Continuous Cost Function Variables

The methodologies and results discussed within this chapter of the thesis have previously been published in (Smith, Benardos and Branson, 2020). The content of this chapter was presented in the journal article in a simplified form whilst here it is presented within the context of the overall research approach.

4.1. Introduction

In Chapters 2 and 3, a need was identified for dynamic cost functions to use changes in online production data to update knowledge of a worker's current performance or capabilities. It is proposed to use continuous variables in these dynamic cost functions to assess a worker's performance as it gradually changes across a work shift. These variables utilise continuous production data obtained from human and robot workers to detect small gradual changes in worker performance across a work shift. Such continuous production data is collected from each iteration of a manufacturing subtask and assessed against the manufacturer's requirements and/or the workers performance in previous work shifts for the manufacturing subtasks. This provides a valuable source of data to quantify a worker's capabilities, allowing quantification of their performance against the manufacturer's expectations whilst also enabling inference of changes in their capabilities when their performance has deviated from nominal. The former capability of these continuous variables allows comparison between workers in factors such as subtask completion times, whilst the later capability allows the inference of the onset of factors such as fatigue that affect performance of human workers.

This chapter begins with the development of two examples of continuous variables that can be implemented within the proposed dynamic cost functions. First in Section 4.2, a fatigue variable is proposed to infer relative fatigue levels through the difference between current completion times and expected completion times for a human worker. Second in Section 4.3, a completion variable is proposed to quantify the standard of worker completion times. These variables are tested with a real robot worker and a simulated worker as a part of dynamic cost functions. To achieve this, it is first necessary to discuss the experimental setup for the robot worker and the setup of simulations for a human worker in the two example subtasks in Section 4.4. Following this it is possible to present and discuss the generated test data to assess the ability of the continuous variables to quantify the capabilities of human and robot workers.

4.2. Continuous Variable Example – Fatigue

4.2.1. Human Fatigue and its Relation to Continuous Variables

Fatigue represents a key factor that affects human performance and capabilities when working over long periods of time, and often manifests itself in two distinct forms that have different effects on a human worker. The first form of fatigue, physical fatigue, can result in reduced physical performance leading to increased subtask completion times. The second form of fatigue, mental fatigue, can result in reduced cognitive abilities such as concentration leading to possible mistakes in tasks (Dawson et al., 2011). Due to this it is desirable to minimise the level of fatigue for human workers to ensure that they can complete tasks effectively and minimise production errors. Fatigue represents an appropriate test case for continuous variables as the onset of both forms of fatigue is gradual over long time periods, however, absolute quantification of fatigue levels poses a significant research challenge.

Although its effects can be measured, a measure of fatigue level itself is often considered as an abstract concept rather than something that can be definitively measured. Previous research has attempted to directly detect physical fatigue through the use of EMG sensors placed on the muscles of a human worker (Potvin and Bent, 1997). However, this could prove difficult to scale to multiple workers due to the infrastructure and processing required to collect and analyse this data. Additionally, this could prove invasive for human workers due to the collection of personal health data and would possibly be rejected by workers and pose ethical questions for its use. In other research, modelling has instead been used to predict or quantify the level of fatigue that will be experienced by a worker using data relating to the applied manufacturing task as shown in Chapter 2. Such existing models use factors such as workload and time spent completing the task to determine fatigue of the human worker. In other models, it was instead proposed to predict the effect of fatigue on workers performance in completing tasks using historic data of previous performance instead of attempting to quantify the level of fatigue itself. This has been done for repetitive manufacturing tasks by generating models that predict the increase in completion times with iterations of a repetitive task based on historic data from previous work shifts. All of these models face the limitation that they cannot react to unexpected events in a human worker's life that are unpredictable yet could severely affect their performance in completing tasks.

In this research, it is instead proposed to attempt to detect an increase in the level of fatigue as it is occurring, to ensure that the fatigue variable accurately represents the level of fatigue the human worker is experiencing during the current work shift. This is achieved by quantifying relative fatigue instead of defining an absolute fatigue value for a human worker by using

existing models combined with real time production data. This relative level of fatigue is inferred by comparing current completion times of a subtask for a human worker with their nominal expected completion times for each subtask of the overall manufacturing task. To define the fatigue variable in this way, a frame of reference given by a baseline model is required to define the nominal human completion times for a subtask across a work shift.

4.2.2. Baseline Model – The Digiesi Completion Time Model

Previous research, as stated in Section 4.2.1, has identified models that relate the effect of fatigue on completion times to the number of iterations of the task completed, validating the inference of the level of fatigue based on the rate of completion time increase used in the fatigue variable. In (Digiesi et al., 2009), a model was presented that is capable of modelling the effects of fatigue and learning phenomena for a human worker on the completion times of a repetitive task over numerous iterations of the task. This model was validated against worker data from a real-world automotive assembly plant, where it was found that in cognitive tasks the fatigue phenomenon prevailed over the learning phenomenon. The model was then approximated in (Digiesi et al., 2009) to remove the learning factor, for situations where the fatigue phenomenon prevails over the learning phenomenon. The approximated model proposed by (Digiesi et al., 2009) gives the completion time, $E_{i,j}$, for a human worker as

$$E_{i,j} = t_{w,j} + \tau'_j \ln(i) \tag{4.1}$$

in the i^{th} task iteration of a subtask j with initial completion time, $t_{w,j}$, and synthetic measure of fatigue, τ'_j. For the completion time to be calculated using the model, historic data for the human worker are required. First, an initial completion time, $t_{w,j}$, is required from the first completed iteration of the subtask in the current task assignment period from which the model evolves. Since in this research, a subtask can be taken away from a worker and be reassigned to them again later, it is necessary to specify the initial completion time from task iteration, w, where

$$w = \begin{cases} 1 \quad if \;\; t_{i,j} \neq 0 \;\; for\, all \;\; i \in \mathbb{Z}^* \\ s+1 \qquad if\; s = \max(i) \;|t_{i,j} = 0 \end{cases} \tag{4.2}$$

to ensure the initial completion time, $t_{w,j}$, is from the current assignment period of the j^{th} subtask to a human worker. Secondly, historic data from previous work shifts are required to calculate the synthetic measure of the fatigue phenomenon, τ'_j, used by the model. This synthetic measure of fatigue was defined in (Digiesi et al., 2009) by the limit

$$\tau'_j \leq \frac{T_j - N_j \cdot t_{w,j}}{N_j \cdot \ln(N_j) - N_j} \tag{4.3}$$

for the i^{th} task iteration of a subtask j where N_j is the number of task iterations completed in a task assignment period of length T_j seconds. In this research the upper limit is used for the synthetic measure of fatigue, given by

$$\tau'_j = \frac{T_j - N_j \cdot t_{w,j}}{N_j \cdot \ln(N_j) - N_j} \tag{4.4}$$

to provide the worst-case scenario of expected completion time change due to the fatigue level for a human worker.

Using the model for human completion times in a repetitive task given by Eq. (4.1), the fatigue variable, $f_{1,j}$, in a subtask j is defined based on the difference between current and expected human completion times for a subtask. In this research, it is assumed that the level of fatigue presented by this model is unavoidable and represents the natural way in which a human will become fatigued as they complete a repetitive manufacturing subtask. This is reasonable to assume because a human worker's performance will naturally decline over a work shift, despite the level of fatigue they are experiencing. When the human worker's completion times are as predicted by Eq. (4.1), this nominal performance infers they have a natural level of fatigue and the fatigue variable should remain at zero. If completion times increase from this level of nominal performance, this worse than expected performance infers that the worker's relative fatigue level is "over fatigued" and that they might have been assigned too much work. If completion times instead decrease from this level of nominal performance, this better than expected performance infers that the worker's relative fatigue level is "under fatigued" and that they might be underutilised. Before defining how the fatigue variable should increase or decrease with these changing levels of fatigue, it is necessary to define a tolerance to ensure the fatigue variable does not increase or decrease unfairly due to natural variations in human performance.

4.2.3. Tolerances to Insignificant Completion Time Variation

It is widely accepted that human completion times are not consistent and are subject to a natural variance that increases in magnitude with the magnitude of completion times. Such a small variation in completion times should thus be considered insignificant and not affect the output cost of the fatigue variable as it is assumed this does not infer a change in fatigue levels of the human worker. To account for this natural insignificant variation in completion times, two methods are implemented to ensure the fatigue variable is unaffected by insignificant changes in a human worker's completion times. First, the fatigue variable compares the expected completion from Eq. (4.1) with a moving average of a set number of the most recent completion times, $r_{i,j}$, for the i^{th} task iteration of a subtask j. A moving average is used to calculate the output cost of the fatigue variable, $f_{1,j}$, since using individual

completion times would cause the fatigue variable to change far too rapidly due to the inconsistency in human completion times. This would also unfairly represent a worker for unusually high or low completion times due to singular outliers over a task assignment period which are not representative of their overall current performance.

Second, to mitigate for the natural variation in human completion times it is also required to set a difference tolerance region to allow a proportionally small variation from the exact expected completion time before the fatigue variable is affected. This is required as in reality when performing nominally, a human worker's completion times will not be exactly the same as the expected completion times predicted by Eq. (4.1) due to the natural variance in human completion times. To this end, a human worker should not be allocated an unrepresentative output cost for the fatigue variable when the difference between actual and expected completion times is proportionally small compared to the expected completion time given by Eq. (4.1). To define this difference tolerance region to variation in completion times for the fatigue variable, it is first necessary to define a tolerance of h_j seconds, for a subtask j, of which human completion times can deviate from the model. This tolerance will have to be set for each specific subtask based on the subtask length. In this research, tolerances will be assigned by hand for the experiments in this chapter, however, in an operational Human-Robot task planning system it would be desirable for the tolerance to be set through autonomous generic methods.

Using the tolerance to variation in completion times, h_j, it is possible to define the upper and lower limit completion times of the difference tolerance region as

$$R_+ = E_{i,j} + h_j \tag{4.5}$$
$$R_- = E_{i,j} - h_j, \tag{4.6}$$

respectively, for the i^{th} task iteration of a subtask j. This enables the definition that if the moving average completion time, $r_{i,j}$, for the i^{th} task iteration of a subtask j exists within the region [R_-, R_+] then the output cost of the fatigue variable, $f_{1,j}$ should remain at zero. Outside of this tolerance region the output cost must increase or decrease to a maximum or minimum limit based on the difference between the moving average completion time for a worker and the expected completion time given by Eq. (4.1).

4.2.4. Effect of Relative Performance on Cost and Definition of Fatigue Variable

To define the output cost of the fatigue variable when the worker's performance deviates from nominal, it is necessary to define maximum and minimum limits for completion times that represent when the output cost

should reach its maximum or minimum value. This enables the fatigue variable to be normalised in a range of [-1, 1] as with the other continuous cost function variable in addition to providing a limit for the maximum acceptable change in a worker's completion times. To define these limits, input from the manufacturer is necessary to define a maximum acceptable percentage, e_j, increase or decrease in completion times from the expected completion time $E_{i,j}$ for the i^{th} task iteration of a subtask j. Using this maximum acceptable percentage, the completion times that result in the fatigue variable reaching its absolute maximum value are defined by

$$\Omega_+ = E_{i,j}\left(1 + \frac{e_j}{100}\right) + h_j \tag{4.7}$$

$$\Omega_- = E_{i,j}\left(1 - \frac{e_j}{100}\right) - h_j, \tag{4.8}$$

respectively, for the i^{th} task iteration of a subtask j. Between these limits and the upper and lower limits of the tolerance region the cost should increase or decrease linearly to its maximum or minimum point. Collating the limits defined by Eqs. (4.5), (4.6), (4.7) and (4.8) with this linear progression allows the fatigue variable to be defined as

$$f_{1,j}(r_{i,j}) = \begin{cases} 1 \quad if \quad r_{i,j} > \Omega_+ \\ \frac{r_{i,j} - R_+}{\Omega_+ - R_+} \quad if \quad \Omega_+ \geq r_{i,j} > R_+ \\ 0 \quad if \quad R_+ \geq r_{i,j} \geq R_- \\ -\left(\frac{R_- - r_{i,j}}{R_- - \Omega_-}\right) \quad if \quad R_- > r_{i,j} \geq \Omega_- \\ -1 \quad if \quad r_{i,j} < \Omega_- \end{cases} \tag{4.9}$$

for the i^{th} task iteration of a subtask j.

4.3. Continuous Variable Example – Completion Times

It is also proposed to use a continuous variable to quantify the standard of completion times for human and robot workers completing a manufacturing subtask. Subtask completion times are chosen as an additional continuous variable as this allows a direct point of comparison between human and robot workers in their ability to meet the performance specifications for a manufacturing subtask. This is permissible, despite the dependency this creates with the Fatigue variable, as the Fatigue variable uses this to define cost for relative fatigue based on the difference between expected and current completion times. However, the completion variable defines a cost based on the magnitude of the completion times themselves and represents a different aspect of human performance. Although such variables are inherently simple by nature, they form a core that the dynamic cost functions can be built around by allowing a direct point of comparison between human and robot workers. These core variables can then be used in conjunction with other variables that

are specific to human or robot workers, such as the fatigue variable. This enables the cost from the dynamic cost function to accurately reflect a worker's capabilities whilst retaining a base commonality between the cost functions for workers.

To define this completion time cost function variable, it is first necessary to provide context as to what is a good or bad completion time for the manufacturing subtask in question. In an industrial manufacturing environment, a manufacturer would expect a product to be completed within a predetermined production time which can be broken down into work element times for each subtask. These work element times give a suitable level of context for the completion variable as it allows completion times to be evaluated based on the manufacturer's requirements. It is assumed that the manufacturer implementing the system will provide a list of the desired work element times for each subtask of an overall manufacturing task with the task specifications.

As with the fatigue variable, it is necessary to set maximum and minimum limits for completion times to allow the output cost of the variable to be normalised over a range of [-1, 1]. To calculate the output cost of the variable, the same moving average of a set number of the most recent completion times, $r_{i,j}$, is used as that in the fatigue variable. This is again utilised to ensure a smooth stable change in output cost and reduce the susceptibility of the variable to outliers in completion times that could unfairly represent the capabilities of a worker. It is desirable to use the same moving average to provide a commonality between the variables and ensure changes in completion time affect the variable at the same rate. This enables the completion variable to be defined as

$$f_{2,j}(r_{i,j}) = \begin{cases} \dfrac{r_{i,j} - H_j}{H_j} & if \;\; 0 \leq r_{i,j} \leq 2H_j \\ 1 & if \;\; 2H_j < r_{i,j} \end{cases} \tag{4.10}$$

for the i^{th} task iteration of a subtask j, where H_j is the manufacturer's desired work element time for the subtask.

4.4. Continuous Variables Testing and Results

4.4.1. Assembly Subtasks for Testing

The Fatigue and Completion variables proposed in this research are tested together within a single cost function for two example manufacturing tasks representing subtasks of a possible overall manufacturing task. These two example tasks are used to illustrate cases where one of the workers should be better suited to the subtask being analysed, with one illustrating a case where the use of a human worker is more appropriate and the other where use of a

robot worker is more appropriate. The first task consists of tightening a 3D printed bolt into a threaded hole over its entire length, the bolt used is an M15 bolt with a thread length of 40mm and a pitch of 2mm. This primitive task represents a case where the use of a human worker is more appropriate, as their increased dexterity and motion speed allow them to complete the task faster than a robot worker could. To complete the task, the worker must pick up the bolt from a holder and screw its entire length into a fixing. The execution time measured is only the time taken to tighten the bolt into the fixing. The experimental setup for this primitive task can be seen in the image given in Figure 4.1 for a robot worker.

Figure 4.1: Experimental setup for the bolt tightening subtask.

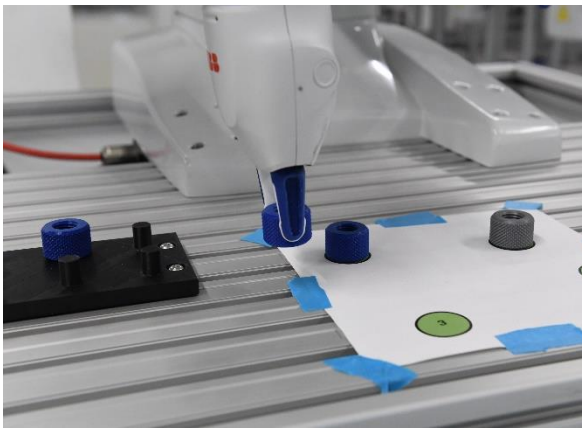

Figure 4.2: Experimental setup for the pick and place subtask

The second example task is a simple pick and place task, requiring a worker to pick up four 3D printed nuts from a holder in sequential order and place each nut in one of four predefined placement positions. The task complexity is increased by randomly selecting the placement position for each nut from one of the four predefined placement positions, simulating high mix production by using the same production techniques but with changing specifications. This example task represents a case where the use of a robot worker is more appropriate, as a robot worker can quickly retrieve instructions for each task iteration and follow them with high accuracy. In comparison, a human worker must check the task specifications before executing it, then verify the task has been completed correctly once finished, increasing their completion time. Additionally, with the cognitive elements of the task such as checking the required task specifications, fatigue can have a large impact on the human workers performance and lead to mistakes in task execution. The

experimental setup for this example task can be seen in the image given by Figure 4.2.

These example tasks are executed by an ABB YuMi robot (ABB, 2020) to obtain completion times from a typical cobot for testing the cost functions consisting of continuous variables. This cobot is used to generate test data since its high level of human-safe features allows close working with a human worker. Additionally, this robot has a high precision of movement and is specifically designed to work in industrial manufacturing environments. To complete the two example tasks, the robot is programmed using the Robot Operating System (ROS) interface developed by Berkeley Automation (Liang, 2016). Completion times for the human worker are then simulated using the model given by Eqs. (4.1) and (4.4) based on various simulated initial completion times and simulated historic data. It is decided to simulate the human worker's completion times in this way to enable illustration of cases where a human worker is considered as "over fatigued" or "under fatigued" in addition to performing as expected for large numbers of task iterations. To make these simulated human completion times more realistic, a random variable generated from the standard normal distribution is added to each simulated completion time to simulate the natural variation in human completion times. The standard normal distribution is chosen as it is assumed that the variation of human completion times from the model given by Eqs. (4.1) and (4.4) is at most 3 seconds, due to the short length of completion times for a human worker in such tasks. Over the next two subsections the methods for generating the completion times for the robot worker and simulated human worker are described for each example subtask. In addition to this, the parameters required to formulate the cost functions, consisting of the fatigue and completion variables, for the workers in the two example subtasks are defined. These completion times and cost functions are evaluated within the Matlab software package for both the bolt tightening and pick and place example subtasks.

4.4.2. Parameter Setting: Subtask 1 Bolt Tightening

First, to generate completion times for the robot worker, the ABB YuMi cobot is programmed to complete 15 iterations of the bolt tightening task and completion times are recorded between the robot starting to tighten the bolt in the fixing and the completion of the tightening of the bolt.

To evaluate the cost function variables for the robot worker, it is also necessary to define the parameters required for the variables to function. For this primitive subtask, only the completion variable, $f_{2,j}$, is used since the fatigue variable, $f_{1,j}$, is not applicable to robot workers. To use the completion variable, it is necessary to set the work element time, H_j, as 45 seconds since it is assumed this would be set based on the fastest possible human completion

time. This is used as the human worker is faster than the robot worker to complete this subtask which would be reflected in the manufacturer's desired work element time.

To calculate the cost for the robot worker to complete the subtask using the dynamic cost function, it is necessary to set the weightings for the variables by utilising the schema detailed in Section 3.6. For this primitive subtask, since the fatigue variable, $f_{1,j}$, is irrelevant to the robot worker it receives a weighting of zero whilst the remaining completion variable, $f_{2,j}$, receives a weighting of one in the dynamic cost function.

Second, the corresponding completion times for a typical human worker are simulated for 15 iterations of the subtask using the model given by Eqs. (4.1) and (4.4). Three sets of initial conditions, given in Table 4.1, are used to illustrate the three potential levels of fatigue for the human worker where they were performing as expected in addition to when they would be considered as over fatigued or under fatigued. These initial conditions provide the necessary data for the model, consisting of an initial completion time for the work shift in addition to the number task iterations a worker can complete over an hour-long time period. These data are estimated for a typical skilled human worker based around the time taken to tighten the bolt in this test, which is determined to be around 47 seconds for a human worker. It is assumed that the initial completion time would only vary by a few seconds when the worker is over or under fatigued due to the relatively short length of time that the task takes. However, even a small change in the initial completion time could result in a large variation in the number of task iterations completed over the work shift if the worker is over fatigued or under fatigued and this is reflected in the simulated initial conditions.

Table 4.1: Initial conditions used for the simulated human worker in the bolt tightening subtask.

Behaviour of Human Worker	**Initial Completion Time (seconds)**	**Number of Task Iterations Completed**
Over Fatigued	50	64
Under Fatigued	45	75
As Expected	47	70
Historical	47	70

To evaluate the cost function variables for the human worker, it is also necessary to define parameters required to calculate the cost function variables for the human worker. For the fatigue cost function variable, $f_{1,j}$, it is necessary to set the baseline expected completion times, $E_{i,j}$, using Eqs. (4.1) and (4.4). The initial conditions used for the simulated worker performing as expected are again used as such performance occurs when the human worker is performing as the historical data would suggest. Next, the tolerance to variation in completion times, h_j, is set at 3 seconds as it is assumed to be

sufficient for a task of such a short length. The final parameter required for the variable, e_j, is set at 20% to allow Eqs. (4.7) and (4.8) to determine the increase or decrease in completion times from the worker's expected completion times that should result in the output cost for the fatigue variable reaching its maximum or minimum value. The percentage is set at this value as it is assumed that a manufacturer would be less tolerant to decline in performance of the human worker for such a simple task. To use the completion variable for the human worker, it is necessary to set the work element time, H_j, as the same value used by the robot worker.

To calculate the cost for the human worker to complete the subtask using the dynamic cost function, it is necessary to set the weightings for the variables by again utilising the schema detailed in Section 3.6. For this primitive subtask, both the fatigue variable, $f_{1,j}$, and the completion variable, $f_{2,j}$, are considered equally important and so both receive a weighting of 0.5 in the dynamic cost function.

4.4.3. Parameter Setting: Subtask 2 Pick and Place

To obtain the input data for the dynamic cost functions for the pick and place task it is first necessary to generate completion times for the robot worker. Here, the ABB YuMi cobot is programmed to complete 90 iterations of the pick and place task with completion times recorded between the robot moving to pick up the first nut and placing the final nut.

Following this, it is necessary to define the parameters required to evaluate the cost function variables for the robot worker. In this subtask only the completion variable, $f_{2,j}$, is again used since the fatigue variable, $f_{1,j}$, is not applicable to robot workers. To apply the completion variable, the work element time, H_j, is set at 27 seconds since it is assumed this would be set based on the fastest possible worker completion time which is from the robot worker in this subtask.

To calculate the complete dynamic cost function for the robot worker in this subtask, it is again necessary to set the weightings for the variables by utilising the schema detailed in Section 3.6. For this example task, the fatigue variable, $f_{1,j}$, again receives a weighting of zero since it is irrelevant to the robot worker whilst the remaining completion variable, $f_{2,j}$, receives a weighting of one in the dynamic cost function.

Table 4.2: Initial conditions used for the simulated human worker in the pick and place subtask.

Behaviour of Human Worker	Initial Completion Time (seconds)	Number of Task Iterations Completed
Over Fatigued	51	61
Under Fatigued	45	74
As Expected	48	69
Historical	48	69

To obtain the input data for the dynamic cost functions, it is next necessary to generate the completion times for a typical human worker to complete the 90 iterations of the pick and place task using the model given by Eqs. (4.1) and (4.4). Three sets of initial conditions, given in Table 4.2, are again used to illustrate the three potential levels of fatigue for the human worker where they are performing as expected in addition to when they would be considered as over fatigued or under fatigued. These data are again estimated for a typical skilled human worker based around the time taken to complete the task, which is determined to be around 48 seconds.

To evaluate the cost function variables for the human worker, it is again necessary to define parameters required to calculate the cost function variables for the human worker. For the fatigue cost function variable, $f_{1,j}$, the baseline expected completion times, $E_{i,j}$, are again set using Eqs. (4.1) and (4.4). The initial conditions used for the simulated worker performing as expected are again used as such performance occurs when the human worker is performing as the historical data would suggest. Next, the tolerance to variation in completion times, h_j, and the variable, e_j, are set to the same values used in the bolt tightening task due to the similarity in the initial human completion times for each task. To use the completion variable for the human worker, it is again necessary to set the work element time, H_j, as the same value used by the robot worker.

To calculate the cost for the human worker to complete the subtask using the dynamic cost function, it is necessary to set the weightings for the variables by again utilising the schema detailed in Section 3.6. The pick and place subtask can be broken down into three primitive tasks, these include checking the placement location of each nut, moving each nut to its position and checking the placement locations are correct. In this subtask the completion variable, $f_{2,j}$, only affects the physical elements of the task since the time taken to complete the cognitive elements of the task can be considered as negligible. In contrast, the fatigue variable, $f_{1,j}$, affects all elements of the task since an over fatigued worker may make mistakes with the cognitive elements of tasks and complete the physical elements more slowly. Utilising the schema given in Section 3.6, the fatigue variable, $f_{1,j}$, is

given a weighting of 0.75 and the completion variable, $f_{2,j}$, is given a weighting of 0.25.

4.4.4. Results – Bolt Tightening Subtask

Given the settings and data generated in Section 4.4.2 the total cost for the robot worker across the 15 iterations of the bolt tightening subtask is given in Figure 4.3 with the output cost of the completion variable being given in Figure 4.4. In this subtask, the total cost for the robot worker to complete the subtask immediately reaches a cost of one and remains at this across all the task iterations simulated. This demonstrates that the robot worker is poorly suited to the task as it achieves the maximum attainable cost for the subtask for the entirety of the task assignment period. This is due to the completion cost function variable which also immediately reaches a maximum cost of one and is maintained across the task assignment period as seen in Figure 4.4. This behaviour of the cost function variable is caused by the long completion times for the robot compared to the work element time for the task since the robot lacked the high dexterity and motion speed within its wrist joint required to complete this task quickly.

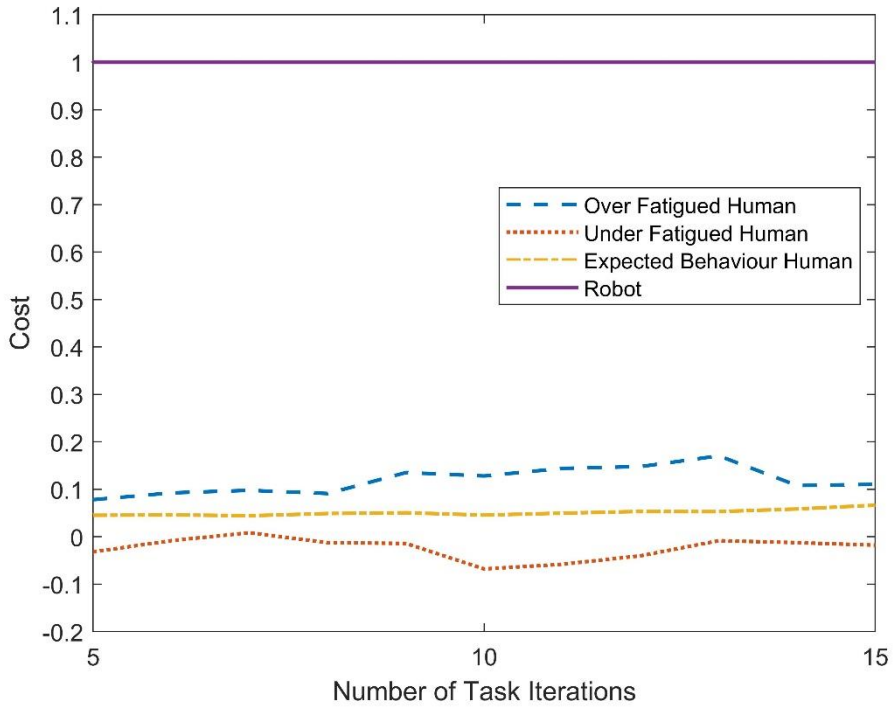

Figure 4.3: Total costs for workers to complete the bolt tightening subtask.

The total cost for the human worker in the bolt tightening subtask can also be seen in Figure 4.3, with the output cost for the completion and fatigue cost function variables shown in Figure 4.4 and Figure 4.5, respectively. In this subtask the human has a mean cost of 0.0511 over the task assignment period when performing as expected with a maximum cost of 0.0665 at the 15th

iteration of the subtask and a minimum cost of 0.0443 at the 7th iteration of the subtask. This total cost for completion of the subtask can be solely attributed to the completion variable as Figure 4.5 shows that the fatigue variable remains at zero for the entirety of the task assignment period for this scenario of human performance. The completion variable is shown to have the same behaviour as the total cost in this subtask except the magnitude is double that of the total cost, as seen in Figure 4.4, due to the weighting of this variable within the cost function for the human worker. This indicates that a human worker performing as expected should not have a large increase in cost over a task assignment period as human completion times naturally increase with fatigue. This also demonstrates that when the worker is behaving as expected, that the cost for the worker to complete the task should only be dependent on the difference between their completion times and the manufacturer's expectations.

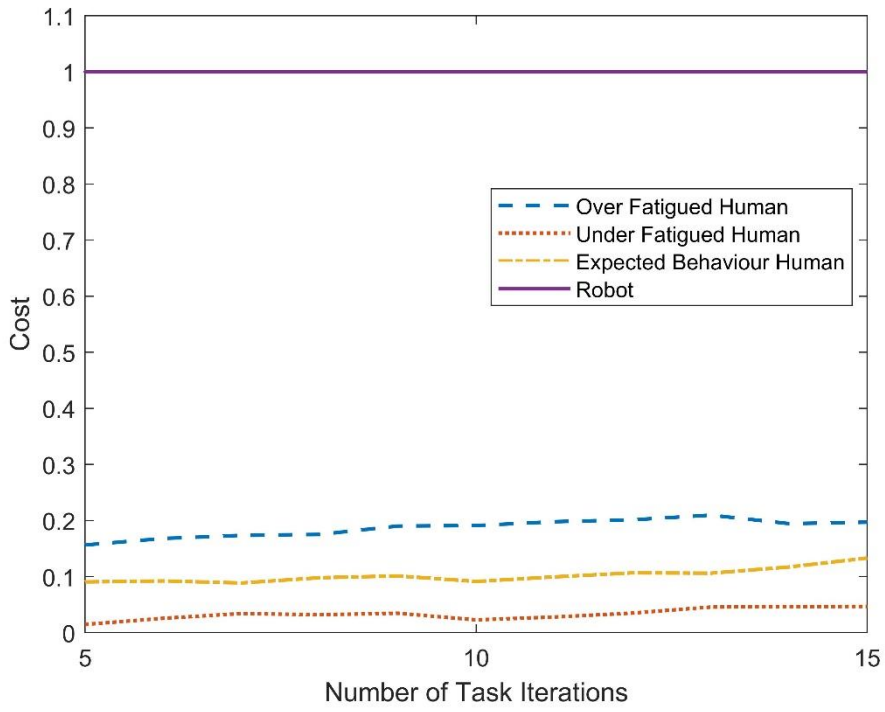

Figure 4.4: Completion cost function variable output costs for workers in the bolt tightening subtask.

Next in the scenario of the over fatigued human worker, Figure 4.3 shows that their mean total cost to complete the subtask is 0.1186 over the task assignment period, with a maximum cost of 0.1705 during the 13th iteration of the subtask and a minimum cost of 0.0782 during the 5th iteration of the subtask. In comparison with the worker performing as expected, Figure 4.3 shows that the over fatigued worker's total cost increases gradually over the task assignment period, however, the total cost behaves noticeably more unexpectedly at certain points. This is most noticeably seen as a significant increase in total cost between the 8th and 9th iterations of the task and a

significant decrease in cost between the 13th and 14th iterations of the task. Examining the output of the constituent cost function variables during the task assignment period shows that the completion variable increases steadily in cost over the majority of the task assignment period from an initial cost of 0.1564 to a final cost of 0.1970 as seen in Figure 4.4. In comparison, the fatigue variable behaves more erratically with the variable producing a minimal output cost until the 9th iteration of the task where it increases significantly followed by a gentle increase until the 13th task iteration where the output cost significantly decreases again. This period between the 9th and 13th iteration, where the fatigue variable shows a greater magnitude of approximately 0.1 can be seen to affect the total cost with an increase in cost over the same period.

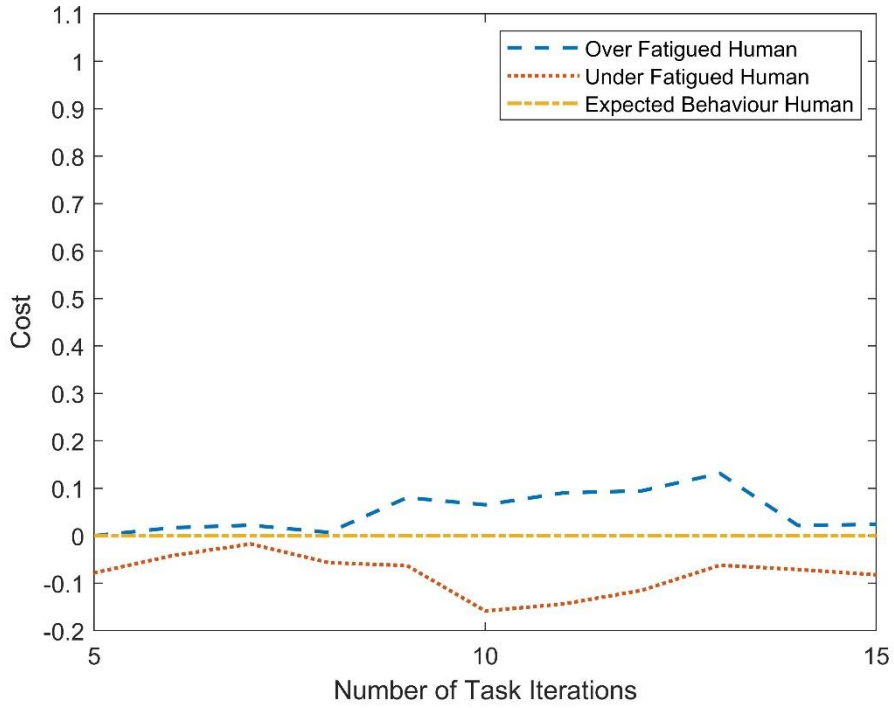

Figure 4.5: Fatigue cost function variable output costs for workers in the bolt tightening subtask.

Finally in the scenario of the under fatigued human worker, Figure 4.3 shows that their mean total cost to complete the subtask is -0.0238, with a maximum cost of 0.0084 during the 7th iteration of the subtask and minimum cost of -0.0679 during the 10th iteration of the subtask. In comparison with the worker in the other fatigue scenarios, Figure 4.3 shows that the under fatigued worker's total cost to complete the subtask behaves far more erratically across the task assignment period with no overall obvious increasing or decreasing trend in the total cost. Despite this, the total cost increases minimally overall from -0.0318 during the 5th iteration of the subtask to -0.0177 during the 15th iteration of the subtask. Determining the cause of this behaviour by examining the output cost of the constituent cost function variables, it is shown that the completion variable has a gentle overall increase in output cost from 0.0146

during the 5th iteration of the subtask to 0.0468 during the 15th iteration of the subtask with a mean cost of 0.0332. The output cost of the fatigue variable instead shows a similar pattern of increase and decrease in cost to that of the total cost for the cost function during the task assignment period. The fatigue variable has a mean output cost -0.0809 during the task assignment period with a maximum cost of -0.0172 during the 7th iteration of the subtask and a minimum cost of -0.1586 during the 10th iteration of the subtask. The greater magnitude of the output cost of the fatigue variable in this task assignment period causes its behaviour to be the dominant influence on the behaviour of the total cost for the subtask generated by the cost function. This can be best observed between the 10th and 12th iterations of the subtask, where the magnitude of the fatigue variable is so large compared to the completion variable that it defines the total cost for the worker to complete the subtask resulting in a period of significant negative cost. The only instance where the completion variable exerts such an influence on the total cost defined by the cost function is during the 7th iteration of the task assignment period where it has a greater magnitude than the fatigue variable, resulting in the maximum total cost for the worker to complete the subtask over the task assignment period.

Now comparing the total cost for the workers in this test case, Figure 4.3 shows that the most suitable worker for the task is the human worker regardless of the level of fatigue they are experiencing. Here, the high completion times for the robot worker result in them attaining the maximum possible cost to complete the subtask showing their unsuitability for assignment of the task. Although the simulated human worker has a fairly low cost to complete the subtask regardless of their level of fatigue, there are clear distinctions between the total costs for them to complete the subtask for the various fatigue levels.

4.4.5. Results – Pick and Place Subtask

Examining the results for the pick and place subtask, the total cost for the robot worker across the 90 iterations of the subtask is given in Figure 4.6 with the output cost of the completion variable being given in Figure 4.7. In this subtask, the total cost for the robot worker has a mean cost of 0.051 over the task assignment period with a maximum cost of 0.0686 during the 88th task iteration and a minimum cost of 0.0278 during the 39th task iteration. Since the completion variable is the sole variable used in the robot worker's cost function it also provides the total output cost for the worker as shown in Figure 4.7. This behaviour in the total output cost and completion variable can be seen since the robot worker can complete the task close to the manufacturers specified completion time. The variance in the completion time variable in this case can be attributed to the change in placement location for each nut in each iteration

of the subtask resulting in different robot arm trajectories for each iteration and thus different completion times.

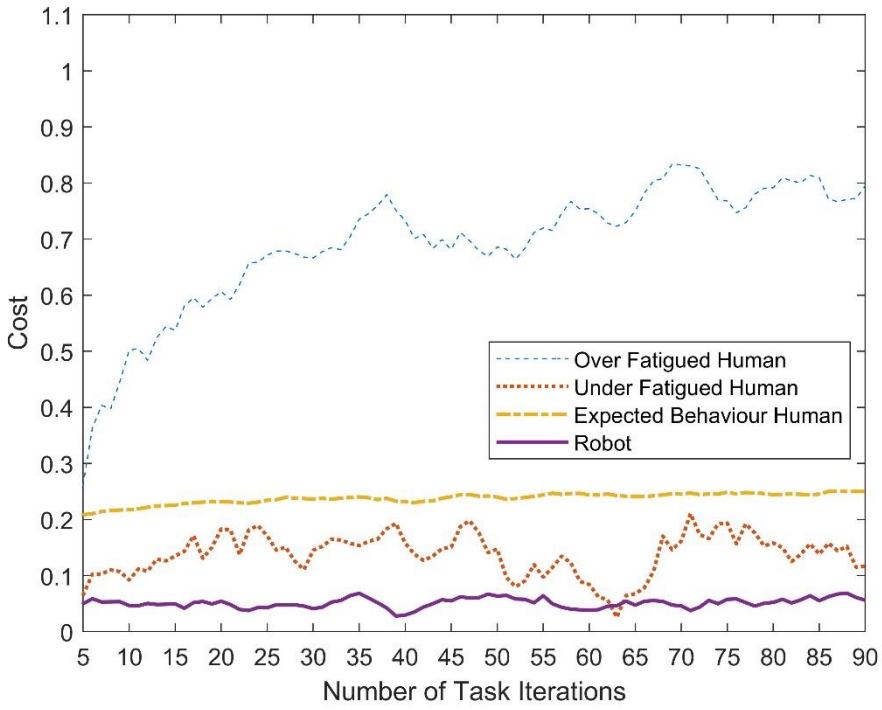

Figure 4.6: Total costs for workers to complete the pick and place subtask.

In the pick and place subtask, Figure 4.6 shows the total cost generated by the cost function for the simulated human worker to complete the subtask with the output costs of the completion and fatigue variables being shown in Figure 4.7 and Figure 4.8, respectively. For the scenario of the human worker performing as expected, Figure 4.6 shows that their mean total cost to complete the subtask is 0.2377 over the task assignment period, with a maximum cost of 0.25 during the 86th iteration of the subtask and a minimum cost of 0.2085 during the 5th iteration of the subtask. A degradation in performance is more evident over the larger task assignment period of this subtask as seen in Figure 4.6, however, this increase in cost is very gentle over a large number of task iterations. Figure 4.8 shows that when the human worker is performing as expected, the output cost of the fatigue variable remains at zero meaning that the completion variable is the only variable contributing to the total cost. Figure 4.7 shows the completion variable has a much larger magnitude than the total cost with a much more significant cost increase and greater variability. The mean output cost of the completion variable over the task assignment period is 0.9510 with a maximum cost of one during the 86th iteration and a minimum cost of 0.834 during the 5th iteration of the task. Despite the large magnitude of the completion variable which reaches its maximum output cost of 1 towards the end of the task assignment period, the total cost for the human worker to complete the subtask when

performing as expected is much lower due to the weighting of the variable within the cost function. Since the variable receives a weighting of 0.25, its effect on the total cost of the worker to complete the subtask is massively reduced with variability in its behaviour being smoothed. This again indicates that a human worker performing as expected should not have a large increase in cost over a task assignment period as human completion times naturally increase with fatigue. Additionally, this again demonstrates that when the worker is performing as expected, that the cost for the worker to complete the task should only be dependent on the difference between their completion times and the manufacturer's expectations.

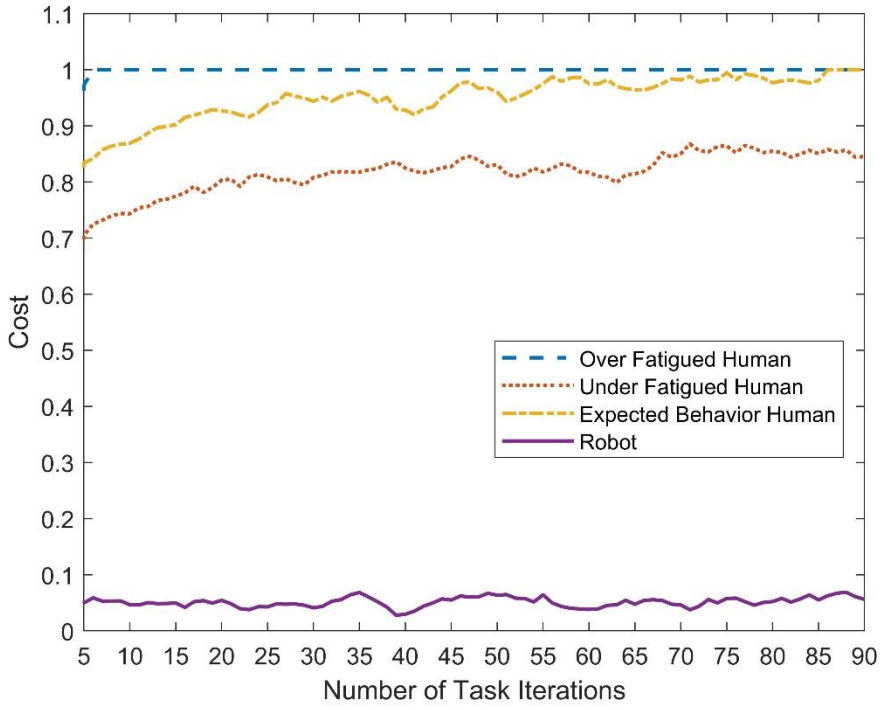

Figure 4.7: Completion cost function variable output costs for workers in the pick and place subtask.

Next in the scenario of the over fatigued human worker, Figure 4.6 shows that their mean total cost to complete the subtask is 0.6903 over the task assignment period, with a maximum cost of 0.8335 during the 69th iteration of the subtask and a minimum cost of 0.2622 during the 5th iteration of the subtask. The over fatigued human worker's cost shows a massive increase over the task assignment period in comparison with the worker performing as expected. Figure 4.6 shows that their total cost to complete the subtask increases rapidly over the first 40 iterations of the subtask with a gentler increase over the remaining iterations of the task assignment period. A significant variation in the total cost of the worker to complete the subtask can also be seen from the 35th iteration of the subtask onwards. Examining the output cost of the constituent cost function variables, Figure 4.7 shows that the completion variable has an output cost of 0.9724 during the 5th iteration of the

subtask which immediately increases to 1 during the 6th iteration of the subtask and remains at this cost for the remainder of the task assignment period. This results in the completion variable making a constant contribution of one weighted at 0.25 to the total cost generated by the cost function for the worker to complete the subtask. In comparison, the fatigue variable displays the same behaviour as the total cost for the worker to complete the subtask but with a lower magnitude as seen in Figure 4.8. This shows the fatigue variable has the dominant effect on the output cost on the behaviour of the total cost for the over fatigued worker to complete the subtask, despite the greater magnitude of the completion variable. This behaviour is seen in the cost function since the cost for the completion variable remains constant for the majority of the task assignment period meaning that it doesn't provide any variability to the total cost and instead only boosts the magnitude of the fatigue variable to give the total cost for the worker to complete the subtask.

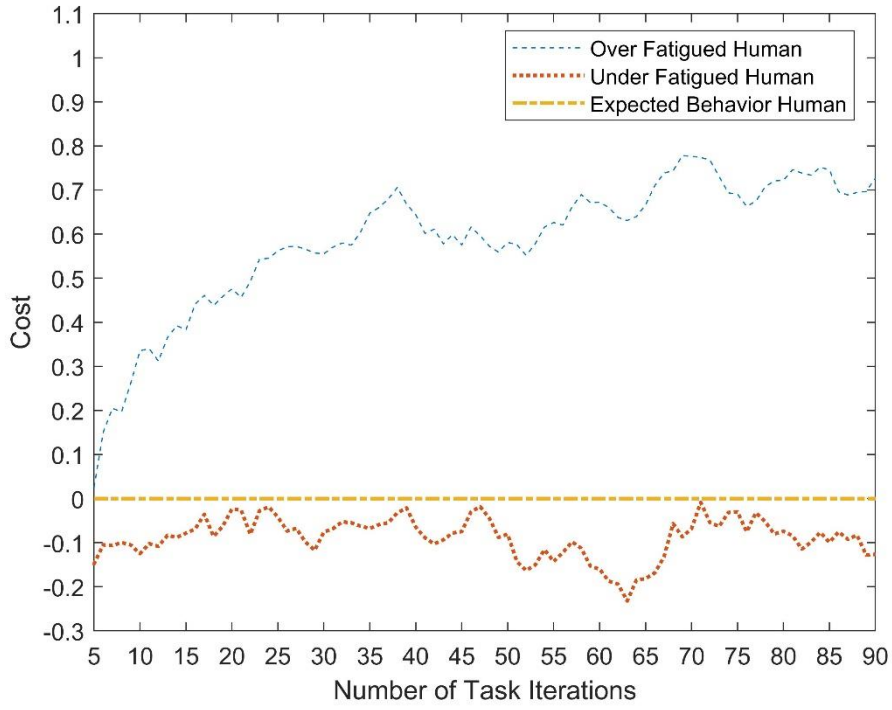

Figure 4.8: Fatigue cost function variable output costs for workers in the pick and place subtask.

Finally in the scenario of the under fatigued human worker, Figure 4.6 shows that their mean total cost to complete the subtask is 0.1377, with a maximum cost of 0.2105 during the 71st iteration of the subtask and minimum cost of 0.0256 during the 63rd iteration of the subtask. Figure 4.6 shows that there was a high variability in the total cost to complete the subtask over the task assignment period which made an overall pattern of change hard to detect as shown with the simulated under fatigued human worker in the bolt tightening subtask. Determining the cause of this behaviour by examining the output cost of the constituent cost function variables, Figure 4.7 shows that the

completion variable has a mean output cost of 0.8169 over the task assignment period with a maximum cost of 0.8679 during the 71st iteration of the subtask and a minimum cost of 0.7051 during the 5th iteration of the subtask. Figure 4.7 shows that the completion variable gently increases in output cost over the task assignment period with none of the unpredictability seen in the behaviour of the total cost to complete the subtask. However, Figure 4.8 shows that this unpredictability can be seen in the output cost of the fatigue variable indicating that the fatigue variable is the dominant influence on the total cost for the worker to complete the subtask across the task assignment period. This occurs since the completion variable shows a gentle increase in cost over the task assignment period, the effect of this is further reduced by the variable's lower weighting of 0.25. In comparison to the completion variable, the fatigue variable has a mean output cost of -0.0887 over the task assignment period, with a maximum cost of -0.0087 during the 71st iteration of the subtask and minimum cost of -0.2321 during the 63rd iteration of the subtask. Figure 4.8 shows that despite the fatigue variable having an output cost that was approximately 10% of that of the completion variable, its higher weighting of 0.75 results in the total cost for the worker to complete the subtask being much lower than the output cost of the completion variable.

Comparing the total cost for the workers in this test case, Figure 4.6 shows that the most suitable worker for the task is the robot worker due to their lower costs over the task assignment period. In this test case, the simulated human worker when under fatigued and performing as expected has a mean total cost of 0.1377 and 0.2377, respectively. In comparison to the bolt tightening task where the robot worker has a much higher cost and is very unsuitable for the task, here the human worker could take over when under fatigued or performing as expected without a huge increase in cost. This is not seen with the simulated over fatigued worker that has a mean cost of 0.6903 over the task assignment period making the worker much more unsuitable in comparison.

4.5. Chapter Summary

In this chapter, continuous cost function variables are proposed to quantify the severity of gradual continuous changes in the performance of workers across a work shift. This is to assess and validate the dynamic cost functions proposed in this thesis. Firstly, two potential continuous cost function variables have been developed to test the concept of the adopted variables. The first of these variables consisted of a fatigue variable that is capable of determining when a human worker's completion times deviated from expected completion times as a measure of fatigue level. The second variable consisted of a completion variable that quantifies the difference between a worker's completion times and the expectations of the manufacturer. Whilst the fatigue variable can only be applied to human workers, the completion variable is

capable of acting as a core variable. A cost function can be built around this variable to ensure that a commonality is retained between cost functions for human and robot workers. This ensures the comparison of costs for workers remains relevant and fair when cost function variables specific to human or robot workers, such as the fatigue variable, are used. These variables have been tested within cost functions for a robot worker and a simulated human worker in two example subtasks where one of the workers is better suited to complete the subtask. These variables were tested by obtaining data from a robot worker completing the task in addition to simulating data for a human worker whilst performing as expected in addition to when they were over or under fatigued.

It was shown in the two example subtasks that the costs generated by the cost functions, containing the specific continuous variables, provided clear distinctions between the robot worker and the simulated human worker under the various fatigue conditions. In both example subtasks it was shown that the total costs for the robot worker and the simulated human worker performing as expected to complete the subtasks progressed in a predictable way over the task assignment period. In the cases of the simulated human worker being under or over fatigued it was instead shown that the total costs for the worker to complete the subtask noticeably deviated from those of the human worker behaving as expected, increasing and decreasing multiple times over the task assignment period. The erratic behaviour of the total cost for the simulated human worker to complete the subtask under these fatigue conditions was shown to be influenced by the fatigue variable thus clearly identifying cases where the human is over fatigued or under fatigued.

In a bolt tightening subtask, it was shown that the robot worker had a significantly larger cost to complete the subtask regardless of the level of fatigue of the simulated human worker indicating that the robot worker was unsuitable to complete the subtask. In the pick and place subtask, it was shown that the robot worker had a lower cost to complete the subtask, however, the cost for the simulated human worker to complete the subtask when they were under fatigued or performing as expected was not significantly larger than that of the robot worker. Due to this it was possible that the human worker could be assigned the subtask under these fatigue conditions without a significant increase in cost. This was shown despite the output cost of the completion variable for the simulated human worker being significantly larger than that of the robot regardless of the level of fatigue the simulated human worker was experiencing. This showed the importance of variable weightings as the heavy weighting of the fatigue variable resulted in a relatively low cost for the simulated worker to perform the task when under fatigued or performing as expected.

The results in this chapter have shown that the proposed fatigue and completion variables provided a way to quantify relative worker performance using continuous production data. A clear distinction between costs generated for workers meaning they can be used to allocate subtasks of an overall manufacturing tasks to workers. However, the effect of the weighting of the continuous cost function variables indicated that additional variables should be used to form a complete cost function. In the next chapter, a discrete variable is proposed to quantify the impact of instantaneous discrete events during production. In contrast to the continuous fatigue and completion variables which quantify worker performance, discrete variables instead quantify a worker's ability to complete a subtask. Combining continuous and discrete variables then creates fully rounded dynamic cost functions that are capable of quantifying a worker's capabilities.

5. Discrete Cost Function Variable

5.1. Introduction

In this research it is proposed to use discrete variables in the dynamic cost functions to quantify the impact of instantaneous discrete events during production. Discrete events, in contrast to the continuous changes in workers performance from Chapter 4, occur during a single task iteration and may have a significant effect on the outcome of a manufacturing task caused by the worker. These discrete events can include production errors or lack of precision in execution of a subtask which result in the failure of the manufacturing subtask, possibly indicating that the subtask should instead be reallocated to another worker. Discrete variables, as with the continuous variables, must be calculated individually for each subtask of the overall manufacturing task to quantify the effect of discrete events on that subtask. This allows assessment of worker capabilities using discrete variables in comparison to the ability of continuous variables to assess performance related factors, such as fatigue or completion time.

This chapter begins with an outline of a generalised structure of a discrete cost function variable in Section 5.2. This includes the operating principle of a discrete variable, which generates costs based on the frequency and severity of discrete events, in addition to the range of output costs of a discrete variable. Next, an example discrete variable is investigated in Section 5.3 for the precision of sealant application a sealant application manufacturing subtask. It is necessary to develop the example discrete variable around as an example subtask as it must be capable of detecting specific errors instead of the generalised input data used by the continuous variables in Chapter 4. Section 5.3 begins with a discussion on sealant pathways in manufacturing followed by a description of the simulated sealant application subtask used in this research. Following this a description is given of the data required to assess the quality of sealant application and the machine vision methods used to obtain this. Next, methodologies are given for grading individual iterations of sealant application utilising this data and a hierarchy of the potential types of error. Methodologies are then given to generate the output cost of the discrete precision of sealant application variable given the frequency and severity of previous errors in addition to reducing the cost when the subtask is completed successfully. Section 5.3 concludes by combining all the methodologies presented into an algorithm defining the discrete variable for precision of sealant application.

Section 5.4 details the testing of the discrete precision of sealant application variable with the simulated sealant application subtask. The variable is utilised in a dynamic cost function, alongside the completion and

fatigue continuous cost function variables given in Chapter 4, to determine its effect on the total cost for workers to complete the sealant application subtask. This is tested by simulating several error scenarios for the human worker to determine the effect on their cost in comparison to that of the robot worker. This also allows analysis of changing worker capabilities on the cost for a worker to complete a subtask and determine if this would result in them no longer being considered the optimal worker to complete the subtask.

5.2. Structure of a Discrete Variable

5.2.1. Operating Principle for Discrete Variables

To develop an example of a discrete cost function variable, it is first necessary to define an operating principle of how such a variable should utilise discrete data. In this research, it is proposed that a discrete variable should grade the output of each iteration of a subtask individually by providing it with a cost. This is required as discrete events will occur in an individual task iteration and not gradually over multiple task iterations as with the continuous variables developed in Chapter 4. In cases where a discrete event has not occurred, a small cost should be generated to determine the quality of task completion between perfect completion and minimum acceptable tolerances. If a discrete error does occur, a higher cost should be generated with this being defined based on the level of severity of the discrete error.

Using the cost for the execution of a single task iteration, the output cost for the discrete variable should be based on the severity and frequency of occurrence of discrete events. To achieve this the output cost should be tolerant to infrequent errors which are natural for a human worker but should increase rapidly if frequent errors occur. This is necessary as frequent errors could imply a change in capabilities for a worker. If the subtask remains assigned to a worker after the discrete variable's cost has increased, the cost should decrease with successfully completed iterations of the subtask. This is necessary as continued successful completion of the subtask implies that the worker's capabilities have not changed.

To allow a discrete variable to generate a cost for each worker for a subtask utilising this methodology, it is proposed to split the model into three elements which enable the core functionality of a discrete variable. These elements are defined as a reaction to small variations in subtask completion, a hierarchy of discrete events and reaction to sudden significant changes in production. In comparison to the continuous variables developed in Chapter 4, discrete variables must be developed around an example subtask as they need to be tuned to the detection of specific errors instead of a generalised form of input data.

5.2.2. Cost Range of a Discrete Variable

To define discrete variables, it is also necessary to define the range of possible output costs in the interval [0,1]. Here an output cost of zero indicates that the worker is capable of completing a subtask perfectly and an output cost of one indicates that a worker has made numerous significant errors completing a subtask indicating it should be reassigned. This decision is taken as when quantifying capabilities, a worker can be capable or incapable of completing a manufacturing subtask thus the cost of the variable should only increase as capabilities change. In comparison the output cost for continuous variables in Chapter 4 was defined in the interval [-1, 1] as workers can perform better or worse than nominal performance so the cost should decrease and increase, respectively, from zero.

5.3. Discrete Variable Example – Precision of Sealant Application

5.3.1. Sealant Pathways in Manufacturing

Sealant pathways are lines in a manufacturing part or product over which a sealant, such as an adhesive, must be applied to create a seal between two parts. Simple easily repeatable tasks can be performed by robot workers. However, more complex sealant pathways with ergonomic difficulties are still often completed by human workers, utilising their skill and craftsmanship (Maiolino et al., 2017). Such tasks require dispensing of a continuous and uniform line of sealant along a specified path to ensure a correct seal has been made. This clear task definition means a discrete change in a task iteration can be easily defined when a task has not been completed within tolerances.

The application of sealant pathways provides a test case to develop a discrete cost function variable. It represents a manufacturing subtask where discrete events caused by a worker can occur in a single iteration of the subtask and lead to failure in its completion. It also represents a manufacturing subtask where small changes are seen in the production data when the subtask has been completed successfully, due to natural variations in worker capabilities. In this example case, it is possible to consider small variations in subtask completion as small deviations in the applied sealant pathways from those specified, where the application is still completed successfully within specified tolerances. This manufacturing subtask also allows the definition of distinct types of discrete event as significant errors in sealant application, which can be ranked into an order of severity. These significant errors occur when the application of sealant by the worker has deviated outside of the specified tolerances resulting in the failure of the manufacturing subtask.

These possible small variations in completion of the subtask and discrete events in a sealant application manufacturing task allow the testing of the complete functionality of the precision of sealant application variable and to demonstrate the possibly of assessing worker capabilities through online production data. Importantly, these small continuous changes and discrete events can also be detected autonomously using machine vision systems without requiring a human supervisor to assess the errors made. Using such technologies, it is possible to extract information about applied sealant pathways from a captured image, allowing characterisation based on a predetermined set of tolerances and a hierarchy of discrete events to determine the severity of individual discrete events. Through these autonomous methods, it is possible to track and characterise discrete events across a work shift and to estimate costs for workers, regarding their capability in the application of sealant through the discrete variable model.

5.3.2. Manufacturing Subtask for Testing

In this example case, a discrete variable is required to assess the quality of application of a straight line of sealant to a workpiece by a human or a robot worker. A straight line was chosen as it represents a simple case where it is possible to easily determine if the subtask has been completed successfully, or if there are any significant deviations between the applied and specified sealant lines. In this example task, the simulated workpiece is a two-dimensional representation of a thin sheet of metal with holes drilled along one side generated in Microsoft Publisher with dimensions: 15 cm long and 4 cm wide as shown in Figure 5.1. This is printed on a sheet of A4 card which represents the workspace for this task. In the simulated sealant application subtask, the specified sealant line should be applied over 13 cm with a width of 0.5 cm. This line should be applied parallel with the edge of the workpiece such that the top left corner of the sealant line is 0.5 cm to the right and 1 cm down from the top left corner of the workpiece as shown in Figure 5.2.

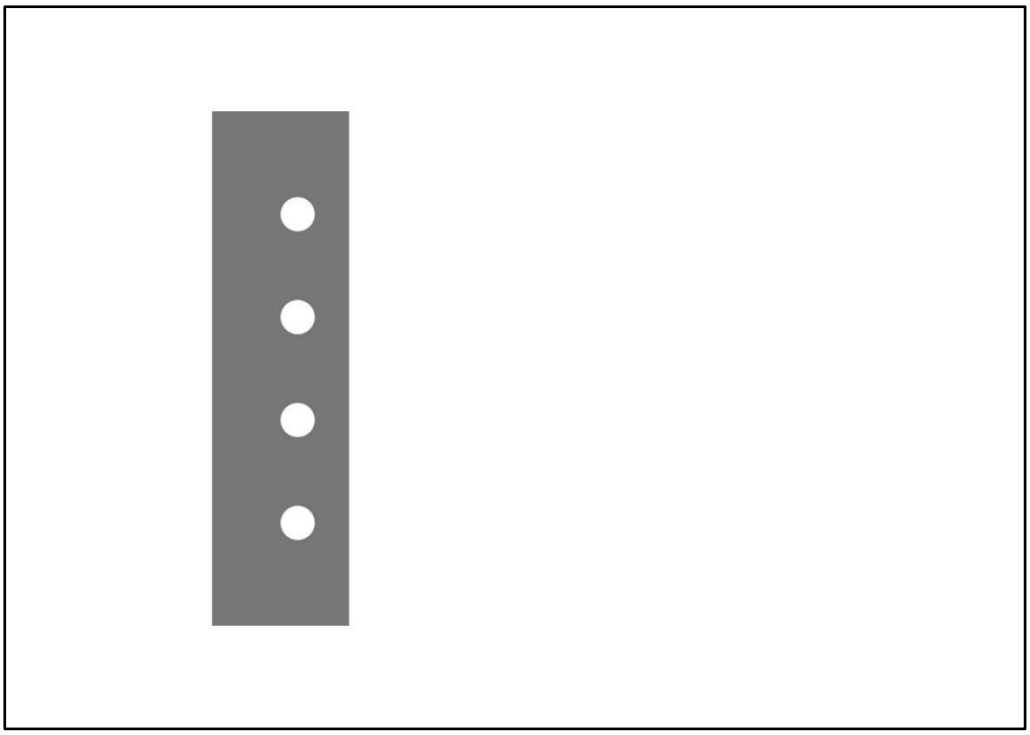

Figure 5.1: The simulated workpiece for the example sealant application subtask.

To simulate sealant application errors for the robot worker, sealant lines are drawn on the simulated workpiece in the drawing package using quadrilateral shapes. This representation is used as robot workers can typically perform tasks with a high degree of accuracy so any variation in a straight line would be minimal. To account for errors in completion of the subtask for a robot worker given their accuracy, it is assumed that these errors would occur due to a failure of hardware or sensor systems detecting where to apply the sealant line. To simulate sealant application errors for the human worker, a clean workpiece is printed, and a simulated sealant line is applied by a human with a black marker pen. This representation is used due to the similarities between sealant applicators and a pen, in terms of the pressure effect applied to the surface on area and the effect of unsteadiness on the quality of application.

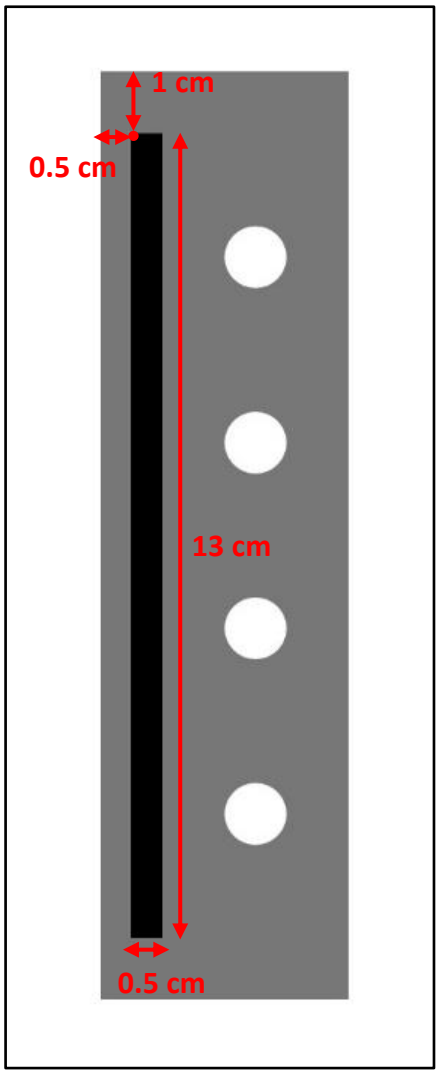

Figure 5.2: A diagram detailing the measurements of the specified sealant line applied to the simulated workpiece.

A simplistic representation of a sealant application task is used in this research because it allows rapid set up and execution. This provides enough data for the required application errors of sealant on a flat surface, representing discrete events in a subtask. In this research, the contribution to knowledge in this chapter is a demonstration of the methodology of reacting to discrete events in online production data for human and robot workers across a work shift given input data regarding the execution of a subtask. Due to this, this example task is considered appropriate as it provides example data without the complex setup of a real-world sealant application task for a human and robot worker. Using this example subtask as a basis, it is next necessary to define the input data required for the proposed discrete variable to assess the quality of sealant application in a single iteration of the subtask and how this input data is obtained.

5.3.3. Input Data Required to Assess Quality

To assess the quality of a straight sealant line applied by a worker, it is first necessary to define the specified sealant line which must be applied and the tolerances that the line must be applied within to be satisfactorily completed. To achieve this, it is necessary to assess the percentage area of the specified sealant line over which sealant is applied, *a*, the percentage length of the specified line over which sealant is applied, *v*, and the angle the applied sealant line deviates from the specified line, *z*. This data is used as the basis for the application of the discrete variable to this manufacturing task as it allows characterisation of the deviation of an applied straight sealant line from a specified sealant line in categories with a definable order of severity. Following this, it is next necessary to define how to obtain this data from a simulated sealant line applied by a human or robot worker.

Figure 5.3: A simulated sealant line for a robot worker on the simulated workpiece with the A4 card representing the workspace and the green screen representing the image background that was isolated.

The input data for the precision of sealant application variable is obtained through machine vision systems using a still image captured of a sealant pathway completed by a worker using a DSLR camera. This is required as the methods presented in this chapter require a high-resolution image to be successful, it is assumed that a manufacturer implementing this system would use a similarly based industrial high-resolution imaging system. In this research, the detection of the simulated workpiece and the sealant pathway applied on it along with its grading, are coded in Python scripts to produce a cost for an individual iteration of the sealant application subtask. To enable this, an image of the A4 print out representing the workspace for the task on a green background is taken as seen in Figure 5.3, this is copied across to the

development PC to allow it to be read by the Python scripts. Once the image is input into the Python scripts, it is first necessary to increase the brightness of the image to aid detection of objects within the image. Using Python libraries developed in (Rosebrock, 2014) based on the open source OpenCV library, it is possible to isolate the workspace from the rest of the original image and transpose the image to the same plane as the workspace to give a top down view. This is achieved by using grayscale conversion and canny edge detection to locate the workspace in the image, followed by image isolation and perspective transform to isolate the workspace from the image. From this image the workpiece is then isolated from the simulated workspace using the same process of brightening the image then applying the isolation and transposition method provided in (Rosebrock, 2014). From this isolated image of the workpiece it is possible to isolate the applied sealant pathway allowing comparison against the specification of a sealant line defined by the manufacturer. To achieve this, it is first necessary to determine how a manufacturer specified sealant line should be defined.

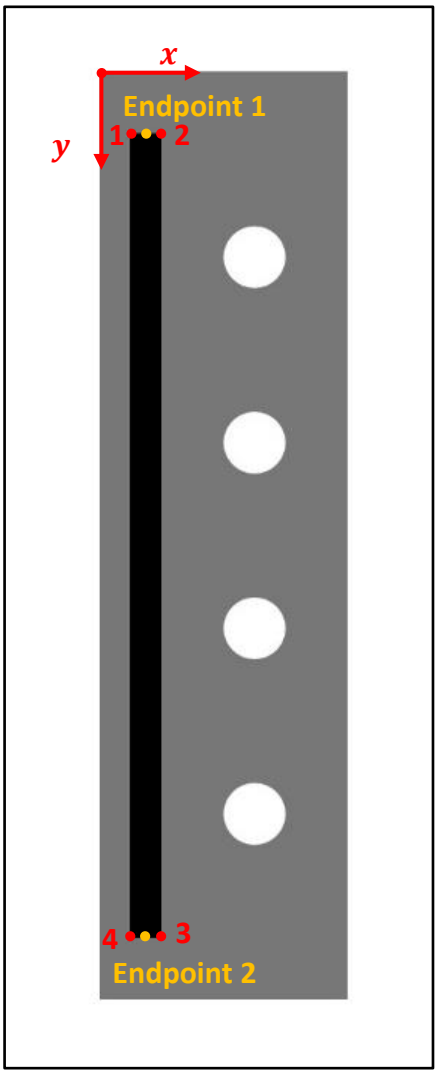

Figure 5.4: A diagram detailing the coordinate system and points required to specify a sealant line for application on the simulated workpiece. This includes four coordinates defining the bounding box of the

sealant line in addition to two endpoints defining the centreline of the specified sealant line for the example sealant application task in this chapter.

In this research, a two-dimensional coordinate system is used to specify sealant lines desired by a manufacturer with sealant lines being specified by the four pairs of coordinates of their vertices (ordered clockwise from the point closest to the origin) and the endpoints of the centreline (endpoint closest to origin given first) given in millimetres. In this coordinate system, the top left of the workpiece is considered as the origin with *x* coordinates progressing across the workpiece towards the right and *y* coordinates progressing down the workpiece as shown in Figure 5.4. This method is used as it follows the same practice as the coordinates of pixels in images, allowing ease of integration with the detection of sealant lines using machine vision. Given the isolated image of the workpiece, it is possible to determine the number of pixels per millimetre in each dimension of the image using the size of the image in pixels and the dimensions of the workpiece in millimetres. This allows the coordinates defining the manufacturer specified sealant line to be transposed into the pixel-based coordinate system of the isolated image of the workpiece to allow comparison between the specified and applied sealant lines.

Figure 5.5: The location of the specified sealant line isolated from the rest of the workpiece when 26% of the area of the specified sealant line is applied by a simulated robot worker.

Figure 5.6: A binary image of the isolated location of the specified sealant line given in Figure 5.2 where the white pixels give the detected black sealant line in Figure 5.5.

To compare the applied and specified sealant lines it is first necessary to determine the percentage area of the specified sealant line over which sealant was applied, *a*. To achieve this the transposed coordinates of the

vertices of the specified sealant line are used to isolate the region of the workpiece where sealant should be from the remainder of the image of the isolated workpiece. This would result in the rectangle defined by points one to four in Figure 5.4 being isolated and extracted for every sealant application in this chapter. An example of this is shown in Figure 5.5 for a simulated sealant application where a robot worker applies 26% of the specified sealant line's area. Following this, the isolated image of the specified sealant line location is colour segmented using the OpenCV Python library to detect pixels of the black applied sealant line located within this region. This binary image is processed using image dilation and erosion to smooth the edges of the locations where the black sealant line was detected to account for any errors in the colour segmentation due to factors such as lighting. Applying this to the isolated image given in Figure 5.5 for the example case where a robot worker applies 26% of the specified sealant line's area results in the binary image given in Figure 5.6. It is then possible to calculate the percentage area of the specified sealant line over which sealant is applied, *a*, using the number of pixels where the applied sealant line is detected and the number of pixels in the image of the location of the specified sealant line.

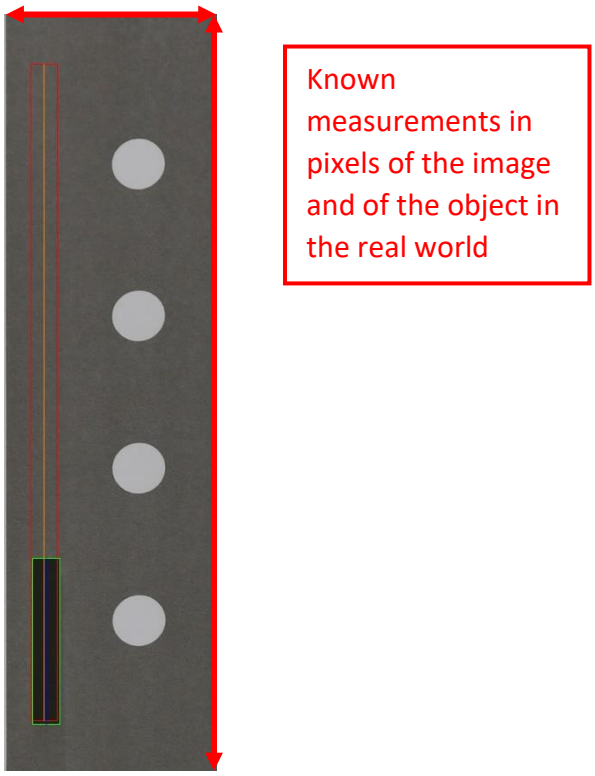

Figure 5.7: An illustration of the dimensions of the workpiece used to generate the values of the number of pixels per millimetre. This allows the coordinates of the vertices of the bounding box of applied sealant lines and the endpoints of the centreline to be transposed to the coordinate system of the image to the real-world coordinate system of the workpiece. The applied sealant line here is a simulated sealant application by a robot worker where 25% of the length of the sealant line is applied. This Figure is adapted from Figure 5.16 which will be seen later in the results section.

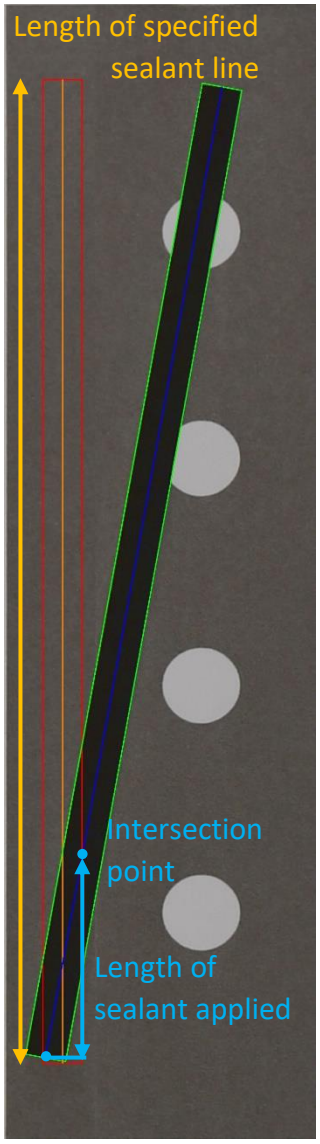

Figure 5.8: An illustration of the methodology for determining the length of a specified sealant line applied when only one endpoint of the applied sealant line lies within the specified sealant line. This is achieved by calculating the intersection point between the centreline of the applied sealant line and the bounding box of the specified sealant line. Utilising the line segment between this intersection point and the endpoint of the applied sealant line with Pythagoras theorem allows the length of sealant applied to be determined. The applied sealant line here is a simulated sealant application by a robot worker where the applied sealant line deviates from the specified sealant line by 10 degrees. This Figure is adapted from Figure 5.17 which will be seen later in the results section.

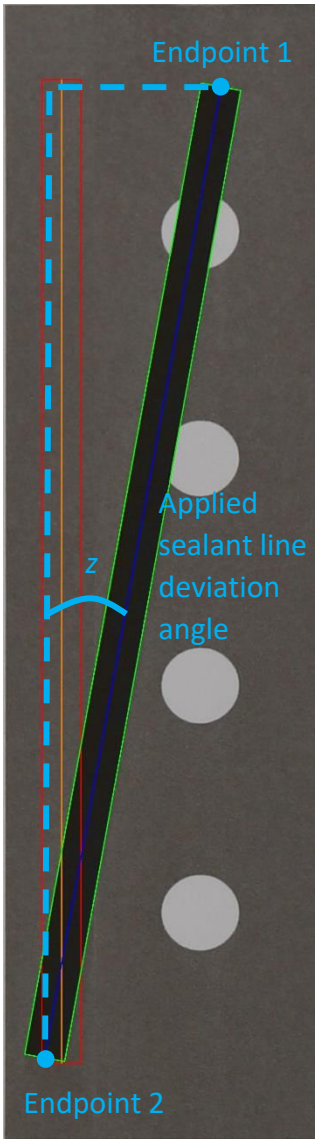

Figure 5.9: An illustration of the methodology for determining the deviation angle between the applied and specified sealant line. This is achieved by determining the angle between the applied sealant line and the y axis since the sealant line should be applied parallel to this. The applied sealant line here is again a simulated sealant application by a robot worker where the applied sealant line deviates from the specified sealant line by 10 degrees. This Figure is adapted from Figure 5.17 which will be seen later in the results section.

It is next necessary to determine the percentage length of the specified line over which sealant was applied, *v*, and the angle the applied sealant line deviated from the specified line, *z*. To achieve this, it is necessary to now colour segment the entire isolated image of the workpiece using the OpenCV Python

library to detect pixels of the black applied sealant line and create a binary image showing the location of applied sealant on the workpiece. Minimum bounding boxes are generated around contours of pixels above a defined threshold size to ensure any image noise is ignored. Using the number of pixels per millimetre in each dimension of the image of the isolated workpiece defined earlier in this section, it is possible to transpose the coordinates of the vertices and endpoints of the centreline of any sealant lines detected into the real world coordinate system defined for the specified sealant line. Since the applied sealant line may be broken into several line segments, this process is completed for each detected segment of a sealant line.

To determine the percentage length of the specified line over which sealant is applied, *v*, it is necessary to use the coordinates of the vertices of the specified sealant line addition to the coordinates of the endpoints of the centrelines of the minimum bounding boxes around any detected applied sealant lines. By generating the linear equations for the four edges of the specified sealant line using its vertices, it is possible to determine if the end points of the applied sealant line are located within the area of the specified sealant line. This approach is applied by modifying the code in (Ruud de Jong, 2017), allowing substitution of the coordinates of the endpoints of the centreline of an applied sealant line into the linear equations of all the edges and determining the sign of the resulting answer. If both endpoints are contained within the specified line, the line length is calculated using Pythagoras theorem. If only one endpoint lies within the specified sealant line, the modified code from (Ruud de Jong, 2017) is again used to determine the intersection point between the centreline of the applied sealant line and the relevant edge of the specified sealant line, with the Pythagorean Theorem then being used between the intersection point and the end point as shown in Figure 5.8. Again, since the applied sealant line may be broken into several line segments, this process is completed for each detected segment of a sealant line and summing their total length to determine the length of the specified sealant line applied and thus the percentage of the length applied to determine *v*.

Finally, to determine the angle the applied sealant line deviates from the specified line, *z*, the end points of the centreline of the detected applied sealant line are used to determine the angle of the line from the y axis of the coordinate system as the specified sealant line is applied vertically down the workpiece as shown in Figure 5.9. Again, since the applied sealant line may be broken into several line segments, this process is completed for each detected segment of a sealant line and the mean angle given as *z*. Using this data, it is possible to assign a cost for an individual iteration of the sealant application subtask.

5.3.4. Reacting to Small Variations in Subtask Completion

In the production data being monitored for discrete events, small variations in completion of the subtask will often be seen even when the task is completed within tolerances. These small variations can relate either to natural variation in a worker's capabilities or could indicate that a discrete event is about to occur in a manufacturing subtask. In either case, these data can provide a valuable insight into the capabilities and performance of a worker and should not be discarded. Small variations in completion of the subtask are considered as changes in production data that are within the tolerances of the manufacturing subtask being executed and so should not have a significant effect on the output cost for the variable. Using this definition, it is necessary to define a set of tolerances to identify whether a change in production data is acceptable or whether a discrete event has occurred. These tolerances are dependent on the specifications of the subtask and must be defined by the manufacturer. These tolerances should not be used to ensure a subtask has been completed perfectly but instead be used to identify if a subtask has been completed adequately to meet production standards.

These acceptable tolerances must be defined in terms of the input data given in Section 5.3.3 including; the percentage area of the specified sealant line over which sealant is applied, the percentage length of the specified line over which sealant is applied and the angle the applied sealant line deviates from the specified line. This tolerance should define the maximum acceptable deviations of an applied sealant line from a specified sealant line that would not cause failure of the subtask. The minimum percentage area and length over which the sealant line must be applied is set at 95% as sealant application tasks require high precision, so it is assumed that a high tolerance is required for completion of the subtask. Due to this, the maximum acceptable angle between the applied and specified sealant lines is set at 2° but this represents an absolute value allowing deviation by this angle to the left or right of this specified sealant line. This tolerance is set with such a small value as even a minor change in angle can result in the majority of the sealant not being applied in the specified location for long sealant lines.

Although these tolerances define the maximum acceptable deviation of an applied sealant line from that specified by the manufacturer, it is necessary to define how the output cost of the discrete variable should increase for a worker between perfect performance and these tolerances. Since the subtask is still completed successfully in these cases, any deviation from perfect performance in the three data types should affect the output cost of the discrete variable equally. This is done since this represents a measure of a worker's capabilities via small changes in production data and any changes in the three types of input data show an equally important deviation from perfect application of sealant by the worker. Given this principle, it is then necessary

to define that the output cost for the discrete variable increases linearly over the cost interval of [0,0.1] with each data type of the input production data providing up to a third of the maximum cost available. Here, 0 represents perfect performance and 0.1 represents the worst performance that is still within the tolerances of the task. This range of small costs is chosen as it ensures that a worker should still be assigned a subtask if they can complete it within the defined tolerances and if performance variables such as the completion variable have a much lower cost than for other workers. This means it is possible to define the output cost of the discrete variable for small continuous changes as

$$\sigma = \frac{0.1}{3}\left(\frac{100-a}{100-95} + \frac{100-v}{100-95} + \frac{|z|}{2}\right). \quad (5.1)$$

Beyond the tolerances defining the maximum acceptable deviation between an applied sealant line and the sealant line specified by the manufacturer, it is necessary to define all major types of discrete events and rank them in order of their severity to identify worker capabilities with the discrete variable.

5.3.5. Hierarchy of Severity of Discrete Events

To understand the significance of a discrete event on the manufacturing subtask it is necessary to characterise discrete events into types and define a hierarchy to quantify the severity of each type of discrete event's effect on the manufacturing subtask. This is important as although a discrete event leads to failure of the manufacturing subtask some may be recoverable, whereas others can necessitate the scrapping of parts and require the subtask to be started over. Within the defined hierarchy of discrete events, each tier should have an associated range of output costs which discrete events of the type belonging to that tier can be given. This hierarchy of discrete events allows a detected discrete event to be characterised into a tier containing discrete events of its type and, based on the range of available output costs for the tier, define an output cost for the discrete event based on its severity within the tier of the hierarchy. Such hierarchies must be defined before production by a manufacturer and are specific to each subtask.

To define this hierarchy of discrete events, it is also necessary to define the range of output costs that the hierarchy of discrete events is defined over as the remaining cost interval of the discrete variable of (0.1,1]. This cost interval is split into equal sections with each section being assigned to a tier of the hierarchy of discrete events in ascending order of severity. Due to this model for the hierarchy of discrete events, it is desirable to limit the number of types of discrete event based on broad general factors that define the event into a type of event and quantify its severity in comparison to other events of that type. If this is not possible it is necessary to ensure costs are given to three

or four decimal places to allow clear definition of discrete events by severity within tiers of the hierarchy of discrete events.

To characterise discrete events of errors in sealant application and rank them, it is first necessary to define the types of discrete event and define an order of severity. Using the production data acting as input to the discrete variable, as described in Section 5.3.3, it is possible to characterise an error in application of sealant into one of the three major error types given in ascending order of significance in Table 5.1. Given that discrete events may cause more than one of these data types to increase above the tolerances, in these cases the data which implies the discrete event with the highest severity has occurred will be used to characterise the discrete event. Using this methodology, it is necessary to define a cost range and method of defining a cost for each type of major sealant application error based on the production data input to the discrete variable.

Table 5.1: The hierarchy of errors for the example sealant application subtask.

Error Severity	**Error**	**Description**	**Data Identifying this Error**	**Cost Range for Errors**
1	Sealant coverage error	Sealant totally covers length of line at correct angle but not enough is applied to fill application area	$a < 95\%$	(0.1,0.4]
2	Sealant gap error	Sealant line is at correct angle but there is a gap in the applied sealant path	$v < 95\%$	(0.4,0.7]
3	Sealant trajectory error	Sealant line has deviated from specified line by significant angle	$\|z\| > 2°$	(0.7,1]

Of these three types of error, the sealant coverage error represents the least significant error. Sealant coverage errors occur when the applied sealant line is applied along the specified trajectory over the entire specified length but, the area of the specified sealant line applied is not within acceptable tolerances. This error is considered the least significant error because the specified length and trajectory are applied implying that the worker can follow the sealant path. However, the application of the incorrect area implies that the worker has reduced capabilities in the techniques required for sealant application. Since the output cost from the hierarchy of discrete events for sealant coverage errors is defined over the interval (0.1, 0.4], the output cost for a sealant coverage error should increase linearly with reduction of the percentage area of the specified sealant line applied, a. This enables the output cost for a sealant coverage error to be defined by

$$\sigma = 0.1 + 0.3\left(\frac{95-a}{95}\right). \quad (5.2)$$

The error with the next level of severity is the sealant gap error which occurs when the sealant line is applied along the specified trajectory but does not cover the entire length of the sealant line specified by the manufacturer. As the specified trajectory is followed, this again implies that the worker can follow the sealant path. However, this error is considered the next most significant as the incorrect length is applied implying that the worker has greater reduced capabilities in the techniques required for sealant application. Since the output cost from the hierarchy of discrete events for sealant gap errors is given in the cost interval (0.4, 0.7], the output cost for a sealant gap error should again increase linearly with reduction of the percentage length of the specified sealant applied, *v*. This enables the output cost for a sealant gap error to be defined by

$$\sigma = 0.4 + 0.3\left(\frac{95-v}{95}\right). \quad (5.3)$$

Finally, the sealant trajectory error represents the most significant type of error and occurs when the sealant line is applied along the wrong trajectory. This error is considered the most significant type of error since the specified trajectory is not followed implying that the worker is not capable of following the specified sealant path and thus has very low capabilities in completing the application of the sealant. As the output cost from the hierarchy of discrete events for sealant trajectory errors is given in the cost interval (0.7, 1], the output cost for a sealant trajectory error should increase linearly with increase of the absolute angle between the specified and applied sealant lines, $|z|$. For this type of error, an upper limit is placed on the value of $|z|$ that defines the maximum cost as it represents an angle and not a not a percentage. Given the severity a small increase in the angle between the specified and applied sealant lines has on the subtask, the cost should increase linearly over the range (0.7,1] proportionately to $|z|$ increasing over the range (2,45]. This enables the output cost for a sealant trajectory error to be defined by

$$\sigma = \begin{cases} 0.7 + 0.3\left(\frac{|z|-2}{45-2}\right) & \text{if } |z| \leq 45° \\ 1 & \text{if } |z| > 45° \end{cases}. \quad (5.4)$$

Once a discrete event is characterised and given a cost based on its severity defined by the hierarchy of discrete events, it is necessary to define the new output cost of the discrete variable based on the severity and frequency of occurrence of previous discrete events.

5.3.6. Reacting to Sudden Significant Changes in Production Data

Although the hierarchy of discrete events allows the severity of a discrete event to be assessed and generate a cost, it is also necessary to intelligently assess factors such as the frequency of occurrences of discrete events to generate the output cost of the discrete variable. This is important as a single occurrence of a discrete event in an isolated case would not necessarily imply that a worker's capabilities have changed and that a task should be reassigned. If after the occurrence of a discrete event the worker continues to complete the manufacturing subtask correctly, this indicates that this was indeed an isolated incident and the worker should continue to execute the subtask. If this discrete event instead continues to occur, this suggests that the worker's capabilities have changed and that the cost from the discrete variable should increase, possibly leading to the reallocation of the manufacturing subtask. This illustrates the need to define "frequency modifier" and "cool down modifier" functions to attenuate the output cost of discrete variables based on the frequency of the occurrences of discrete events and their severity.

Firstly, it is necessary to define a frequency modifier function that generates the output cost of the discrete cost function variable when a discrete event occurs based on the number of previous occurrences of discrete events and their severity over a set number of previous iterations of the sealant application subtask. This frequency modifier function is formed using an exponential growth function where the number of occurrences of discrete events drives the growth and the mean severity of the events defines the magnitude. This is used as discrete variables should be tolerant of one or two discrete events over a large number of subtask iterations, but the output cost should rapidly grow when discrete events repeatedly occur. Using this operating principle, the frequency modifier function, ς, is defined as

$$\varsigma = c\left(e^{\xi\iota} - 1\right) \tag{5.5}$$

where c is the mean cost of previous discrete events, ι is the number of occurrences of discrete events in the past κ number of task iterations and ξ is a constant to attenuate the growth of the frequency modifier.

Here ι and c are autonomously generated but ξ and κ must be set by the manufacturer to determine how tolerant the discrete variable should be to multiple occurrences of discrete events within a set time frame. For the example sealant application subtask, the past κ = 50 iterations of the manufacturing subtask are used to determine the output cost. This represents a significant number of iterations of the sealant task and as such if further errors do not occur in 50 iterations of the subtask then an error can be considered an isolated incident. To set the constant ξ, it is necessary to determine how tolerant the discrete variable should be to the frequency of

previous discrete events. In this example case, it is defined that after three errors have occurred in 50 iterations of the sealant application subtask that the output cost of the discrete variable should be given by the mean cost, *c*, of these error instances given by the hierarchy of discrete events. This is used as one or two errors over 50 iterations could represent isolated incidents but once three errors occur, this implies that worker capabilities appear to be declining. Given this, it is possible to use Eq. (5.5) to define the constant defining the growth of the frequency modifier function as $\xi = \frac{\ln(2)}{3}$.

Secondly, it is necessary to define a cool down modifier function that generates the output cost of the discrete variable following the last occurrence of a discrete event by reducing it as the worker continues to complete the subtask successfully. It is necessary to define this function based on the number of successful iterations of the manufacturing subtask completed since the last discrete event and apply it to generate the output cost for the discrete variable. This cool down modifier function is formed using an exponential decay function where the number of successful task iterations completed since the last discrete event drives the decay. This is used as the discrete variable cost should remain high for the first few successfully completed iterations after a discrete event but once the worker has proven the event was an isolated case, the output cost should decrease rapidly towards nominal levels. This allows the cool down modifier function, Φ, to be defined as

$$\Phi = \upsilon e^{-o\varepsilon} \tag{5.6}$$

where υ is the output cost of the discrete variable after the last occurrence of a discrete event, ε is the number of successfully completed task iterations since the last discrete event and o is a constant to attenuate the decay of the cool down modifier. Here υ and ε are generated autonomously but o must be set by the manufacturer to determine how many iterations of the subtask must be successfully completed by the worker after a discrete event to show that their capability to complete the subtask has not changed.

In this example sealant application subtask, it is defined that if a worker has a current discrete variable output cost of one that completing 20 iterations of the subtask successfully should reduce the discrete variable cost back down to 0.1. This was chosen as it is assumed that this would be sufficient evidence to show that a worker's capabilities have returned to nominal levels. Given this, it is possible to define the constant which attenuates the decay of the cool down modifier as $o = \frac{-\ln(0.1)}{20}$. It is intended that the cool down modifier function should generate the output cost for the discrete variable after a discrete event has occurred until either another discrete event occurs, or the output cost has returned to 0.1. Once the output cost has returned to 0.1 the

discrete variable output cost will once again be defined by reacting to small continuous changes in production data.

5.3.7. Final Formulation of the Discrete Variable

Following the definition of the three core elements that define the operating principle of the discrete variable in Section 5.3.4 to 5.3.6, it is necessary to combine them into a single overall algorithm that defines the discrete variable for the sealant application subtask. This is given in Algorithm 5.1 and provides the output cost of the discrete precision of sealant application variable, $f_{3,i,j}$, in task iteration *i* of subtask *j* given the input data for the variable.

As shown in Algorithm 5.1, the discrete variable must first determine if a discrete event has occurred in the current iteration of the subtask by analysing input production data and determining if the subtask has been completed within defined tolerances. If a discrete event has occurred the variable must determine a cost for the event using the hierarchy of discrete events, then use the frequency modifier function to determine the output cost for the discrete variable. As stated in Sections 5.3.5 and 5.3.6, this allows the discrete variable to react to discrete events in subtask execution assessing their severity individually and then defining an appropriate cost for the worker based on the frequency and severity of any previous discrete events within 50 previous task iterations. If a discrete event has not occurred but a discrete event has occurred within those past 50 iterations of the subtask, the cool down function must be used to generate an appropriate cost for the worker. This cool down function must also be used to determine the output cost for the worker on subsequent iterations of the assembly task, providing another discrete event does not occur, until the output cost from the cool down function drops below 0.1. Once the cool down function has reduced the cost of the discrete variable below 0.1, the output cost must be calculated based on the last iteration of the task completed. As stated in Section 5.3.6, this allows the discrete variable to fairly reduce the output cost for a worker to nominal levels if they continue to perform the manufacturing subtask successfully after a discrete event has occurred. In the cases where no discrete events have occurred within the last 50 iterations of the subtask the output cost for the discrete variable must also be generated in this way.

Algorithm 5.1: Precision of Sealant Application Discrete Variable

Input: percentage area of the specified sealant line over which sealant was applied, a, percentage length of the specified line over which sealant was applied, v, the angle the applied sealant line deviated from the specified line, z, iterations in which discrete events have occurred within the last κ number of task iterations and corresponding output costs from hierarchy of discrete events.

if $a < 95$ and $v \geq 95$ and $|z| \leq 2$

Calculate σ for error using Eq. (5.2)

else if $a < 95$ and $v < 95$ and $|z| \leq 2$

Calculate σ for error using Eq. (5.3)

else if $a < 95$ and $v < 95$ and $|z| > 2$

Calculate σ for error using Eq. (5.4)

else

Calculate σ for error using Eq. (5.1)

end

if $\sigma > 0.1$

Determine output cost of discrete precision of sealant application variable, $f_{3,i,j}$, in task iteration i of subtask j by applying the frequency modifier function given by Eq. (5.5)

else if $\sigma < 0.1$ and $f_{3,i-1,j} > 0.1$

Generate output cost of discrete precision of sealant application variable, $f_{3,i,j}$, using cool down modifier function given by Eq. (5.6)

if output cost of discrete variable is now below 0.1

Generate output cost of discrete precision of sealant application variable, $f_{3,i,j}$,using Eq. (5.1)

end

else

Generate output cost of discrete precision of sealant application variable, $f_{3,i,j}$, using Eq. (5.1)

end

Output: Generated output cost of discrete precision of sealant application variable, $f_{3,i,j}$

5.4. Precision of Sealant Application Variable Testing and Results

5.4.1. Experimental Setup

To test the proposed discrete variable for precision of sealant application, the output of the discrete variable is determined for several simulated scenarios where discrete errors in the sealant application subtask occur across a work shift with different frequencies and severity. To achieve this, it is first necessary to simulate discrete errors of each category in the hierarchy proposed in Section in 5.3.5 for a human and robot worker and to grade them using the hierarchy of discrete events for sealant application. In addition to this it is also necessary to simulate an example case where the human and robot workers have completed the task within the defined tolerances. For the robot worker, the application of sealant was represented by lines being printed with the workpiece on a sheet of A4 card and for the human worker these lines were hand drawn as described in Section 5.3.3.

For the human and robot workers three simulated errors are generated for each of the three levels of error given in the hierarchy of discrete events in Section 5.3.5. First, situations where a sealant coverage error is the most significant error are tested by applying the sealant line over the specified length and trajectory at the specified location but with the incorrect area. Three errors are simulated with 76%, 50% and 26% of the area of the specified line being applied by reducing the width of the line to 3.8 mm, 2.5 mm and 1.3 mm, respectively. Second, situations where a sealant gap error is the most significant error are tested by applying the sealant line with the specified trajectory at the specified location but with the incorrect length. Three errors are again simulated with 75%, 50% and 25% of the length of the specified line being applied by applying 9.75 cm, 6.5 cm and 3.25 cm of the sealant line, respectively. Finally, situations where a sealant trajectory error is the most significant error are tested by applying the sealant line at the specified location but on the wrong trajectory. Three errors were again simulated with the applied sealant line veering to the right of its specified location by 10˚, 15˚ and 20˚. As stated in Section 5.3.2, these errors can be accurately created for the robot worker by simulating the sealant application in the Microsoft Publisher drawing package. However, for a human worker these are hand drawn so are not replicated exactly, which is more in-line with what you would expect in practice. To generate a cost for each of these individual sealant applications, the input data (a, v and z) are first generated in Python scripts from an image of the sealant application as described in Section 5.3.3. Based on these input data, a cost is also generated within Python scripts using either Eq. (5.1), (5.2), (5.3) or (5.4) based on the type of sealant error as described in Algorithm 5.1.

The costs generated for each of the simulated sealant applications are then output and saved in a text file.

Once these sealant application errors have been analysed and given a cost, it is necessary to simulate the discrete variable over several work shifts with different frequencies and severities of errors occurring to demonstrate the response of the discrete error variable. To simulate this, three groups of scenarios of error occurrences are considered that represent situations that could occur for a human worker over 100 iterations of the sealant application task being executed. In this research, these scenarios of error occurrences are simulated for a human worker only with the robot worker completing the sealant application subtask within the defined tolerances throughout the 100 task iterations. This is done as it is assumed that such errors for a robot worker would be unlikely to occur due to their accuracy, and that the occurrence of any such errors would be symptomatic of hardware or software failure which may result in the removal of the robot from production. In each of these simulated scenarios, the errors simulated in the experiments detailed at the start of this section are used with the costs generated for individual errors given in Section 5.4.2. These costs are imported into the MATLAB software package from the text file output by the Python scripts allowing the simulation of the discrete variable and cost functions over a work shift in these scenarios.

First, a group of scenarios are simulated where a human worker makes errors that should not be considered as severe, given by infrequent occurrences of errors which could be considered as isolated instances and do not affect a worker's general capabilities. In these scenarios, two errors in sealant application occur for the human worker in task iterations 15 and 80 of the 100 simulated iterations of the subtask. Three scenarios are considered, in the first scenario the errors in task iterations 15 and 80 will be the 76% area applied and 50% area applied errors, respectively, for the human worker. In the second scenario the errors are the 75% length applied and 50% length applied errors and in the last scenario the errors will be the 20° trajectory deviation and 15° trajectory deviation errors, respectively, for the human worker.

Second, a group of scenarios are considered where the human worker makes frequent errors of the same type when completing the sealant application subtask. These scenarios could occur when a human worker is completing the sealant application subtask rapidly but at the expense of quality of execution of the subtask. Here, frequent occurrences of low severity errors should be considered as mildly severe, however, the frequent occurrence of errors of a higher severity should be considered more significant. In these scenarios, errors in sealant application occur for the human worker every 5 task iterations between iterations 15 and 40 of the 100 simulated iterations of the subtask. Three scenarios are again considered, in the first scenario the errors consisted of 76% area errors in task iterations 15 and 20, 50% area errors in

task iterations 25 and 30 and 26% area errors in task iterations 35 and 40. In the second scenario the errors consisted of 75% length errors in task iterations 15 and 20, 50% length errors in task iterations 25 and 30 and 25% length errors in task iterations 35 and 40. Finally, in the last scenario the errors consisted of 10° trajectory deviation errors in task iterations 15 and 20, 15° trajectory deviation errors in task iterations 25 and 30 and 20° trajectory deviation errors in task iterations 35 and 40.

Third, a single scenario is considered where simulated errors made by a human worker should be considered highly severe, given by the occurrence of errors that increase in frequency and severity over a short period of time. For a human worker, such a scenario could occur when they are experiencing difficulty in completing a subtask due to difficulty of the subtask or possible external factors indicating that their capability in performing the task has greatly reduced. In this scenario, errors in sealant application occur for the human worker every 5 task iterations between iterations 15 and 30 of the 100 simulated iterations of the subtask and consisted of the most severe errors in each tier of the hierarchy of errors. Here, the errors consisted of the 26% area error in task iteration 15, the 25% length error in task iteration 20 and 20° trajectory deviation errors in task iterations 25 and 30.

In all of these groups of scenarios it is necessary to simulate the cost for individual iterations of the sealant application subtask when completed within tolerances for use by the discrete variable, which would be generated using Eq. (5.1) with data from an individual subtask iteration in a real world application of the system. In this research, it is assumed that the human worker has a mean cost of 0.05 in this situation, halfway between perfect performance and the maximum defined tolerance for the sealant application subtask. Given this, their cost when the sealant application subtask is completed within tolerances is simulated via randomly generated variables from an $N(0.05,0.01^2)$ distribution. For the robot worker, it is assumed that costs when the sealant application subtask is completed within tolerances are also normally distributed via the $N(0.02, 0.005^2)$ distribution as their higher accuracy would mean they can apply the sealant pathway closer to perfect application and with less variance in the quality of application. Both of these Normal distributions are truncated between 0 and 0.1 to ensure costs are within the range for workers performing nominally.

In addition to the discrete variable for precision of sealant application it is also necessary to simulate a complete cost function for this subtask to determine the effect of errors on the output cost for such a subtask and when errors would cause a switch in the optimal worker to complete the sealant application subtask. To complete this cost function, the completion variable and fatigue variable, given in Chapter 4, are again used. However, to apply these cost function variables, the parameters used to calculate them must first

be defined. Here, the fatigue variable is given by Eq. (4.9) where the maximum acceptable percentage, e_j, increase or decrease in completion times from the expected completion time $E_{i,j}$ is again set at 20%. A small modification is made to the tolerance of the variable to natural completion time variance by defining this as 5% of the expected completion times, allowing the tolerance to be defined as $h_j = 0.05E_{i,j}$. For the completion variable given by Eq. (4.10), the desired work element time for the task is set at 10.8 seconds which is 90% of the expected completion time assumed for the human worker for this subtask.

Finally, to generate the complete dynamic cost function for the simulated sealant application subtask it is necessary to provide weightings for the dynamic cost functions in addition to the input data for the completion and fatigue variables. In this subtask, sealant application is considered a primitive task itself thus all cost function variables are given an equal weighting for the human and robot workers using the schema in Section 3.6. Here, since all of the variables are relevant to the human worker the discrete precision of sealant application variable along with the fatigue and completion time variables are given a weighting of 1/3. For the robot worker, the completion time and discrete precision of sealant application variables are given an equal weighting of 1/2 since the fatigue variable is not applicable to the robot worker. To generate the completion times for the simulated human worker for the completion and fatigue variables, Digiesi's model (Digiesi et al., 2009) given in Eq. (4.1) is again used. To calculate this, it is assumed that the human worker has an initial completion time of 12 seconds and can complete 225 task iterations over an hour. To simulate the natural variation in human completion times, 5% of the initial human completion time is multiplied by a random variable generated from the N(0,0.125) distribution and added to the completion time generated from the model. In this case it is assumed due to the high accuracy of a robot worker that the robot worker can complete the task with a constant completion time of 18 seconds due to the repetitive nature of the task but is slower than the human worker. As with the example subtasks in Chapter 4, the Fatigue and Completion variables utilise a moving average of the last 5 completion times so costs are given from task iteration 5 onwards Given the complete definition of the simulated sealant application subtask and the input data for the cost functions for the human and robot workers it is possible to observe how the discrete precision of sealant application variable would grade the individual discrete sealant errors described in this section.

5.4.2. Experimental Results – Sealant Grading

Utilising the experiment methodology presented in Section 5.4.1, it is first necessary to observe how the discrete variable grades individual sealant applications using Algorithm 5.1. In the case of the simulated robot worker completing the task within tolerances, Figure 5.10 shows the isolated workpiece with the perceived locations of the specified sealant line,

represented by the red box and the applied sealant line represented by the green box. In this case, a cost of 0.0297 is generated based on the applied sealant line being detected as covering 95.6% of the area and 99.9% of the length of the specified sealant line with no deviation angle between the applied sealant line and the specified sealant line. In this simulated case, the sealant line should have been applied perfectly but an error is seen due to the machine vision methods highlighted in Section 5.3.3. This error occurred as the workpiece is not isolated perfectly resulting in it being offset slightly due to some of the white area of the simulated workspace being included in the isolated image. Although this error is small, the effect on the percentage area of the specified sealant line applied is significant, being reduced to 95.6% instead of 100% since it is perceived that part of the area of the specified sealant line is not applied over the entire length of the specified sealant line as seen in Figure 5.10. Despite this error, the machine vision system was used as the gathering of data on the applied sealant lines is not the focus of the research in this chapter and despite this error, the applied sealant line is graded as within tolerances and given a low cost of 0.0297 due to the other measured variables being close to perfect. If no errors occur for the robot worker this cost would also be the output cost for the discrete precision of sealant application variable and regardless of weighting would have minimal difference on the output cost of the subtask for the robot worker.

Figure 5.10: Isolated simulated workpiece for the robot worker completing the sealant line as specified, with perceived and specified sealant lines highlighted.

Next, it is necessary to observe how the discrete variable grades individual sealant errors for the robot worker, with Table 5.2 showing the costs generated for the robot worker for each error tested in addition to the data *a*, *v* and *z* that defines these costs. Observing the least severe type of error given by sealant coverage errors, Figures 5.11, 5.12 and 5.13 show the isolated workpiece for the errors where the robot worker applies 76%, 50% and 26% of

the area of the specified sealant line, respectively. It is shown that for 76% to 50% of sealant area coverage that the cost of the discrete error still remains quite low in the first quartile of the defined cost range of [0,1] for individual sealant applications. However, the cost increases beyond this into the second quartile for the most serious error of this type. These costs are appropriate as the data collected implies that the worker is still capable of completing the sealant application subtask but is experiencing minor difficulty achieving the accuracy required by the subtask.

Table 5.2: Table of costs for simulated coverage errors for a robot worker and the data that defines the cost of these discrete events.

Error	**Cost for Error Event**	**Percentage of Specified Sealant Line Area Applied, *a***	**Percentage of Specified Sealant Line Length Applied, *v***	**Difference Angle Between Specified and Applied Sealant Lines, *z***
76% Area Applied	0.1573	76.9%	99.8%	0.162°
50% Area Applied	0.2452	49%	100%	0.026°
26% Area Applied	0.3163	26.5%	99.6%	0.192°
75% Length Applied	0.4632	69.2%	75%	0.071°
50% Length Applied	0.5408	48.4%	50.4%	0°
25% Length Applied	0.622	22.6%	24.7%	0°
10° Trajectory Deviation	0.7697	18.7%	20.4%	-10°
15° Trajectory Deviation	0.8035	6.21%	6.18%	-14.8°
20° Trajectory Deviation	0.84	4.85%	4.58%	-20.1°

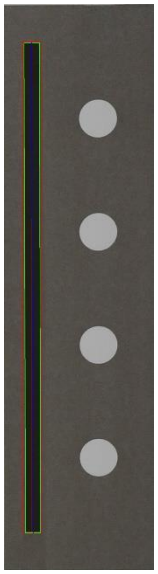

Figure 5.11: Isolated simulated workpiece for the robot worker completing the sealant line with 76% of the specified area applied, perceived and specified sealant lines highlighted.

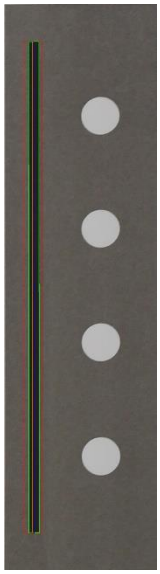

Figure 5.12: Isolated simulated workpiece for the robot worker completing the sealant line with 50% of the specified area applied, perceived and specified sealant lines highlighted.

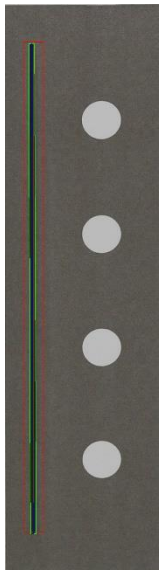

Figure 5.13: Isolated simulated workpiece for the robot worker completing the sealant line with 26% of the specified area applied, perceived and specified sealant lines highlighted.

Observing the next level of severity of error given by sealant gap errors, Figures 5.14, 5.15 and 5.16 show the isolated workpiece for the errors where the robot worker applies 75%, 50% and 25% of the length of the specified sealant line, respectively. Due to the severity of these errors, Table 5.2 shows the cost for each sealant application is significant, being in the second and third quartile of the defined cost range of [0,1]. These costs are again appropriate as the data collected for the sealant application implies that the worker is still capable of following the sealant trajectory but is showing more severe difficulty in the accuracy required by the sealant application subtask.

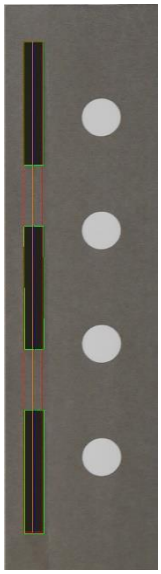

Figure 5.14: Isolated simulated workpiece for the robot worker completing the sealant line with 75% of the specified length applied, perceived and specified sealant lines highlighted.

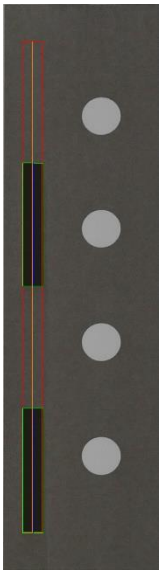

Figure 5.15: Isolated simulated workpiece for the robot worker completing the sealant line with 50% of the specified length applied, perceived and specified sealant lines highlighted.

Figure 5.16: Isolated simulated workpiece for the robot worker completing the sealant line with 25% of the specified length applied, perceived and specified sealant lines highlighted.

Observing the highest level of severity of error given by sealant trajectory errors, Figures 5.17, 5.18 and 5.19 show the isolated workpiece for the errors where the robot worker applies the sealant line on a trajectory that deviated by 10°, 15° and 20° from the specified sealant line, respectively. Table 5.2 shows the largest costs for individual sealant applications can be seen here and are in the fourth quartile of the available cost range of [0,1]. These costs are again appropriate as the data collected for the sealant application implies that the worker has significantly reduced capabilities of following the sealant trajectory that has caused total failure of the sealant application subtask. This is shown in Table 5.2 with the most severe error of a 20° trajectory deviation for the specified sealant line where approximately only 5% of the length and area of the sealant line is applied.

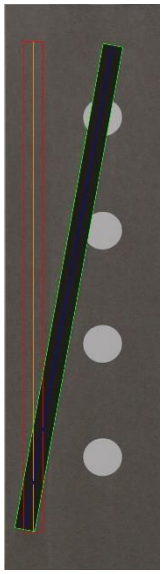

Figure 5.17: Isolated simulated workpiece for the robot worker completing the sealant line with the trajectory deviating by 10°, perceived and specified sealant lines highlighted.

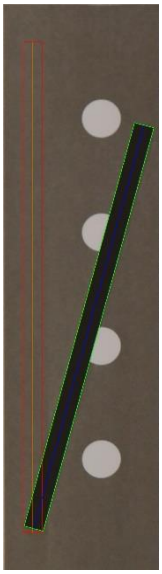

Figure 5.18: Isolated simulated workpiece for the robot worker completing the sealant line with the trajectory deviating by 15°, perceived and specified sealant lines highlighted.

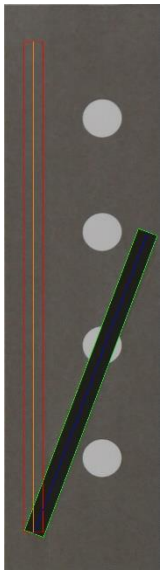

Figure 5.19: Isolated simulated workpiece for the robot worker completing the sealant line with the trajectory deviating by 20°, perceived and specified sealant lines highlighted.

It is next necessary to observe how the discrete variable given in Algorithm 5.1 grades comparable individual sealant applications for the simulated human worker for comparable sealant errors to those simulated for the robot worker. For the simulation of a human worker completing the task within tolerances, Figure 5.20 shows the isolated workpiece with the perceived locations of the specified sealant line, represented by the red box and the applied sealant line represented by the green box. Here, a cost of 0.0245 is generated based on the applied sealant line perceived to be covering 97.9% of the area and 99.2% of the length of the specified sealant line with it deviating from the specified sealant line by 0.302°. Here, a similar cost is given to the robot worker when the sealant is applied within tolerances. However, for the human worker this cost is due to small inaccuracies in the area and length of the applied sealant line, as shown in Figure 5.20, whereas for the robot worker this is due to machine vision errors.

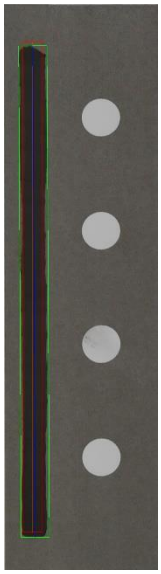

Figure 5.20: Isolated simulated workpiece for the human worker completing the sealant line as specified, with perceived and specified sealant lines highlighted.

Following this, it is necessary to observe how the discrete variable grades individual sealant errors for the simulated human worker, with Table 5.3 showing the costs generated for the robot worker for each error tested in addition to the data *a*, *v* and *z* that defines these costs. First observing the least severe type of error given by sealant coverage errors, similar errors are replicated to those for the robot worker where 76%, 50% and 26% of the area of the specified sealant line was applied. For the human worker, the sealant lines are applied with approximately 68%, 46% and 28% of the specified area covered with Figures 5.21, 5.22 and 5.23, respectively, showing the isolated workpiece. Table 5.3 shows similar costs are given to those of the robot worker for these errors, despite the human worker applying the sealant line with difference angles between the applied and specified sealant lines that are slightly larger but still within acceptable tolerances.

Table 5.3: Table of costs for simulated coverage errors for a human worker and the data that defines the cost of these discrete events.

Error	Cost for Error Event	Percentage of Specified Sealant Line Area Applied, *a*	Percentage of Specified Sealant Line Length Applied, *v*	Difference Angle Between Specified and Applied Sealant Lines, *z*
~68% Area Applied	0.1863	67.7%	100%	0.324°
~46% Area Applied	0.2537	46.3%	100%	0.518°
~28% Area Applied	0.3116	28%	100%	0.517°
~78% Length Applied	0.4549	76.1%	77.6%	1.31°
~53% Length Applied	0.5315	48.1%	53.4%	1.12°
~26% Length Applied	0.6178	22.6%	26%	0.212°
10° Trajectory Deviation	0.7752	17.9%	20.4%	-10°
15° Trajectory Deviation	0.8033	11%	14.1%	-14.8°
20° Trajectory Deviation	0.8381	7.54%	8.23%	-19.8°

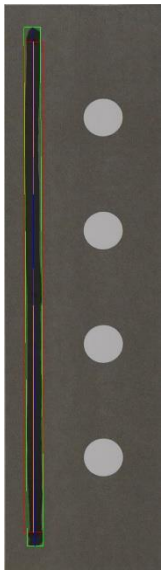

Figure 5.21: Isolated simulated workpiece for the human worker completing the sealant line with approximately 76% of the specified area, perceived and specified sealant lines highlighted.

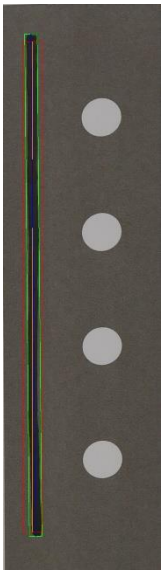

Figure 5.22: Isolated simulated workpiece for the human worker completing the sealant line with approximately 50% of the specified area, perceived and specified sealant lines highlighted.

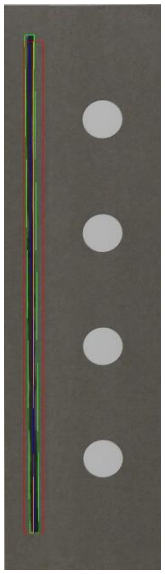

Figure 5.23: Isolated simulated workpiece for the human worker completing the sealant line with approximately 26% of the specified area, perceived and specified sealant lines highlighted.

Observing the next level of severity of error given by sealant gap errors, it is again necessary to replicate similar errors to those of the robot worker where 75%, 50% and 25% of the length of the specified sealant line was applied. For the human worker, the sealant lines are applied with approximately 78%, 53% and 26% of the specified length covered with Figures 5.24, 5.25 and 5.26, respectively, showing the isolated workpiece. Table 5.3 again shows that similar costs are given for these errors to those of the robot worker, despite the human worker applying the sealant line with difference angles between the applied and specified sealant lines that are much larger but still within acceptable tolerances.

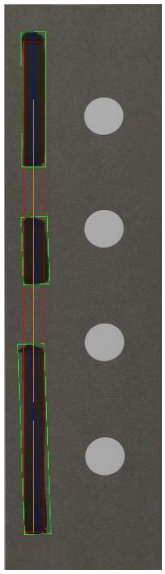

Figure 5.24: Isolated simulated workpiece for the human worker completing the sealant line with approximately 75% of the specified length, perceived and specified sealant lines highlighted.

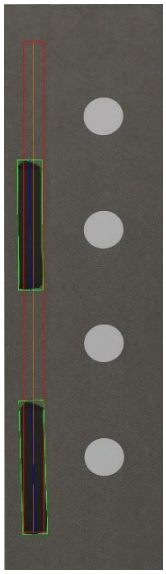

Figure 5.25: Isolated simulated workpiece for the human worker completing the sealant line with approximately 50% of the specified length, perceived and specified sealant lines highlighted.

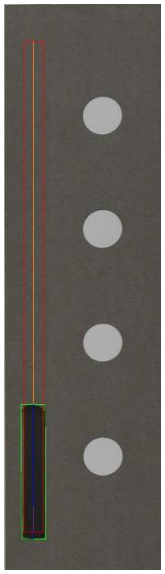

Figure 5.26: Isolated simulated workpiece for the human worker completing the sealant line with approximately 25% of the specified length, perceived and specified sealant lines highlighted.

Finally, it is necessary to observe the highest level of severity of error given by sealant trajectory errors. Here the errors with the robot worker where there are trajectory deviations of 10°, 15° and 20° of the applied the sealant line from the specified sealant were very closely replicated with Figures 5.27, 5.28 and 5.29 show the isolated workpiece, respectively. In these cases where the sealant application errors for the robot worker and human worker are identical for a deviation angle rounded to 3 significant figures, Table 5.3 shows that the cost difference between the human and robot worker for the sealant application instance differs by less than 0.006. This indicated that the discrete variable gives a fair judgement on the capabilities of the human and robot worker for these simulated errors and does not show bias towards a particular worker.

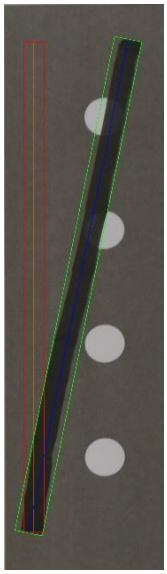

Figure 5.27: Isolated simulated workpiece for the human worker completing the sealant line with the trajectory deviating by approximately 10°, perceived and specified sealant lines highlighted.

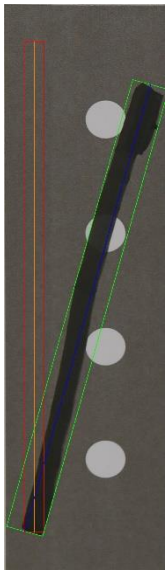

Figure 5.28: Isolated simulated workpiece for the human worker completing the sealant line with the trajectory deviating by approximately 15°, perceived and specified sealant lines highlighted.

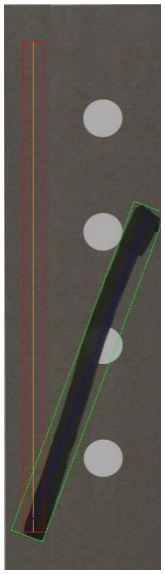

Figure 5.29: Isolated simulated workpiece for the human worker completing the sealant line with the trajectory deviating by approximately 20°, perceived and specified sealant lines highlighted.

5.4.3. Experimental Results – Discrete Variable Simulation Across a Work Shift

Given the costs of sealant application errors discussed in Section 5.4.2, it is necessary to observe how such errors would affect the output cost of the discrete precision of sealant application variable in addition to the total cost for the simulated workers to complete the sealant application subtask in the scenarios described in Section 5.4.1. It is first important to analyse the completion variable for this subtask as seen in Figure 5.30 to determine any effect it would have on the total cost for the human or robot worker to complete the subtask. This shows that the completion variable for the human worker increases gently from a cost of 0.1988 to a cost of 0.5122 across the 100 iterations simulated which is much lower than the constant cost of 0.6667 for the robot worker. Given that the weighting for this variable is 1/2 for the robot worker and 1/3 for the human worker which should have a fatigue variable cost of zero, this gives a maximum cost for the subtask of 0.1707 for the human worker and 0.3334 for the robot worker if they complete the sealant application perfectly. This implies that if the cost for the human worker to complete the sealant application subtask rises above that of the robot worker that this is due to the discrete variable and not the other variables. This assertion applies to each of the scenarios discussed in this section as the same completion time data is used to generate the output cost of the completion variable for the human and robot worker in every scenario tested.

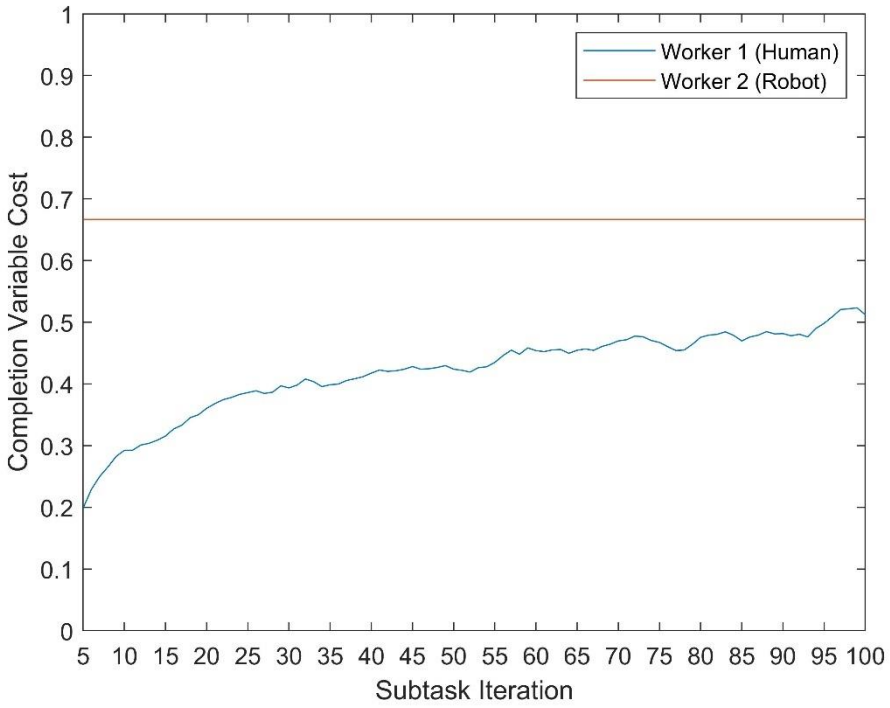

Figure 5.30: A plot of the completion variable cost for workers across the simulated 100 iterations of the sealant application subtask.

For the scenarios described in Section 5.4.1, it is first necessary to observe the group of scenarios where sealant application errors are isolated incidents in task iterations 15 and 80 of the 100 simulated iterations of the subtask. Within this group of scenarios, it is first necessary to analyse the first scenario where these errors are the 76% area applied and 50% area applied errors, respectively, for the human worker. Here, Figure 5.31 shows the output of the discrete precision of sealant application variable and Figure 5.32 shows the output total cost of the subtask given the cost functions described in Section 5.4.1. Figure 5.31 shows that these isolated incidents of sealant application errors did not noticeably affect the cost of the discrete precision of sealant application variable. This demonstrates the tolerance of the discrete precision of sealant application variable to isolated errors, as the output cost of the variable is approximately 26% of the cost of the individual sealant errors. Figure 5.32 showed that this tolerance resulted in the isolated sealant application errors having no noticeable effect on the output total cost for the human worker to complete the subtask. This meant that the human worker had the lowest cost to complete the sealant application subtask across the task iterations despite having a slightly higher discrete precision of sealant application variable cost, since their completion variable cost is much lower than that of the robot worker.

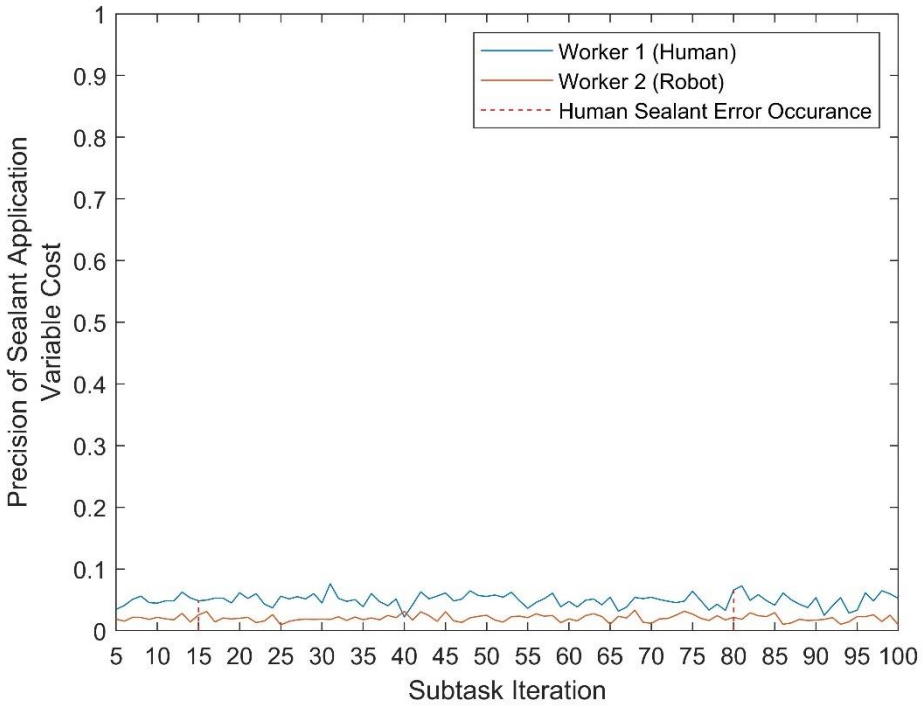

Figure 5.31: Plot of the discrete precision of sealant application variable cost for the human and robot worker in the sealant application subtask with the infrequent occurrence of sealant coverage errors for the human worker.

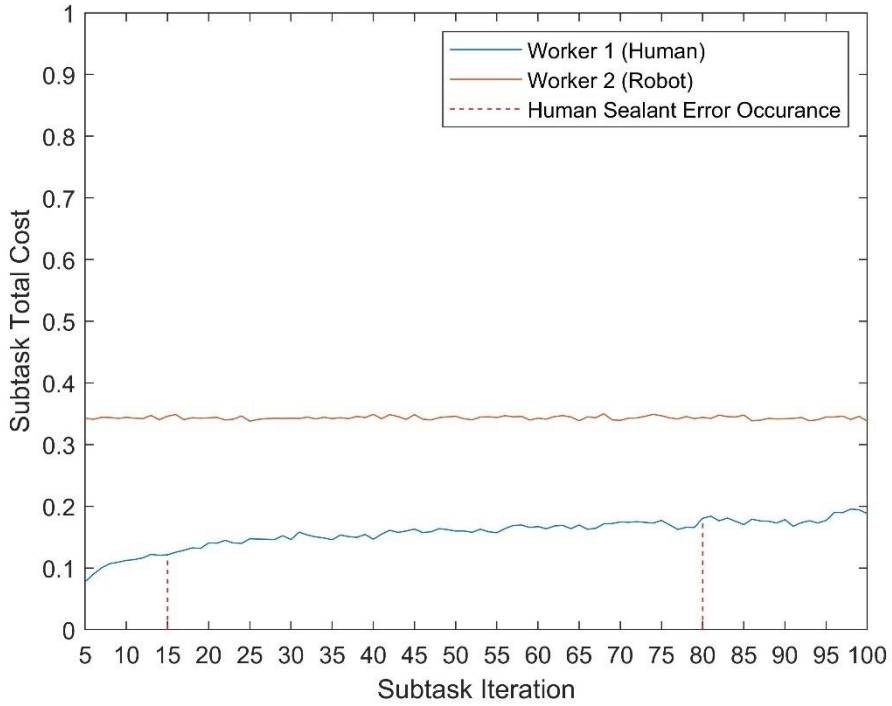

Figure 5.32: Plot of the total cost for the human and robot worker in the sealant application subtask with the infrequent occurrence of sealant coverage errors for the human worker.

Next, for this group of scenarios it is necessary to analyse the second scenario, where the isolated errors at task iterations 15 and 80 are the 75%

length applied and 50% length applied errors, respectively, for the human worker. Here, Figure 5.33 shows the output of the discrete precision of sealant application variable and Figure 5.34 shows the total cost to complete the subtask given the cost functions described in Section 5.4.1. In comparison to the previous scenario, Figure 5.33 shows that the sealant gap errors in this case cause a small increase in the output cost of the discrete precision of sealant application variable to 0.1182 and 0.1381 in task iterations 15 and 80, respectively. However, these are comparatively much lower than the costs for the 75% length applied and 50% length applied errors of 0.4549 and 0.5315, respectively. Once these errors occur in iterations 15 and 80, it takes 2 and 3 iterations, respectively, of the human worker performing the sealant task within tolerances for their costs to return to nominal levels. Figure 5.34 shows that these errors also cause the total cost for the human worker to complete the subtask to increase by 0.065 and 0.1052 in task iterations 15 and 80, respectively, from the cost in the previous iteration. This relatively small increase in total cost meant the human worker still has a much lower cost than the robot worker over the task iterations simulated despite the occurrence of more severe sealant application errors.

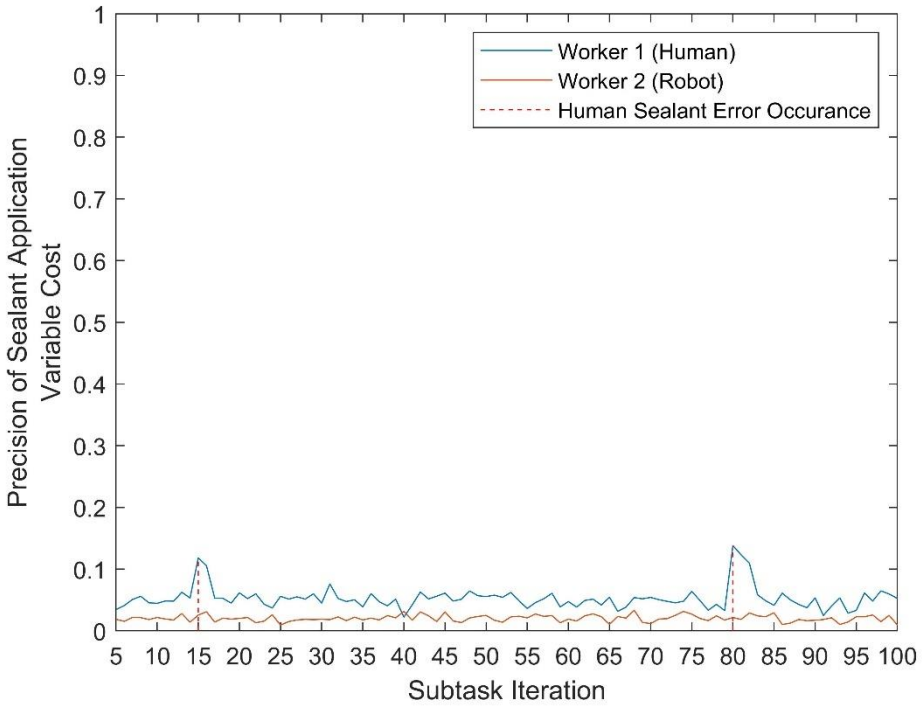

Figure 5.33: Plot of the discrete precision of sealant application variable cost for the human and robot worker in the sealant application subtask with the infrequent occurrence of sealant gap errors for the human worker.

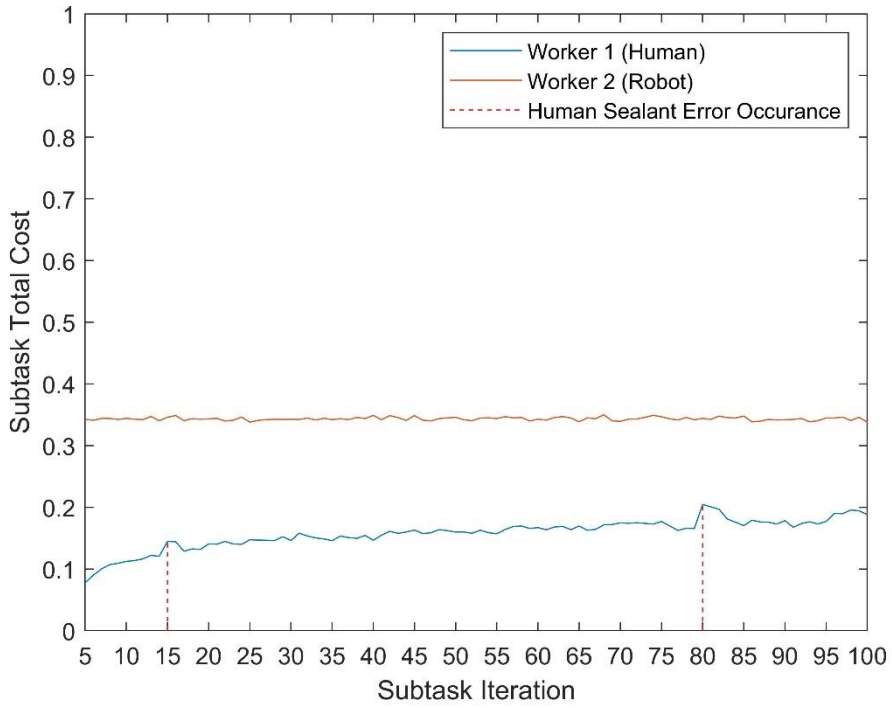

Figure 5.34: Plot of total cost for the human and robot worker in the sealant application subtask with the infrequent occurrence of sealant gap errors for the human worker.

Finally, for this group of scenarios it is necessary to analyse the third scenario where the errors at task iterations 15 and 80 are the 10˚ trajectory deviation and 15˚ trajectory deviation errors, respectively, for the human worker. Here, Figure 5.35 shows the output of the discrete precision of sealant application variable and Figure 5.36 shows the total cost for the workers to complete the subtask given the cost functions described in Section 5.4.1. Figure 5.35 shows that these errors have a much greater effect on the output cost of the discrete precision of sealant application variable, causing the cost for the variable to rise to 0.2178 and 0.2088 in task iterations 15 and 80, respectively. However, these costs are still much lower than the cost for the 10˚ trajectory deviation and 15˚ trajectory deviation errors of 0.7752 and 0.8033, respectively. After these error occurrences at iterations 15 and 80, it takes 7 iterations of the human worker performing the sealant task within tolerances for their costs to return to nominal levels. Figure 5.36 shows that this has a minor effect on the total cost for the human worker to complete the sealant application subtask, raising their total cost in task iterations 15 and 80 by 0.1646 and 0.1759, respectively, from the previous iteration. Although the trajectory deviation errors represent the most severe errors that a worker could cause, Figure 5.36 shows that this is not enough to increase the cost for the human worker to complete the subtask beyond that of the robot worker.

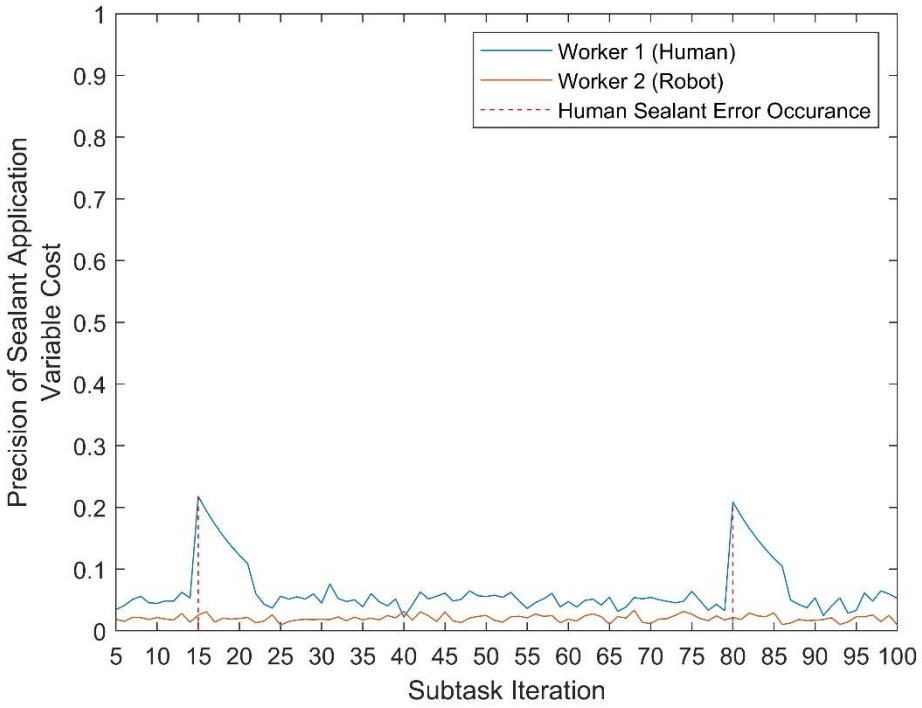

Figure 5.35: Plot of the discrete precision of sealant application variable cost for the human and robot worker in the sealant application subtask with the infrequent occurrence of sealant trajectory errors for the human worker.

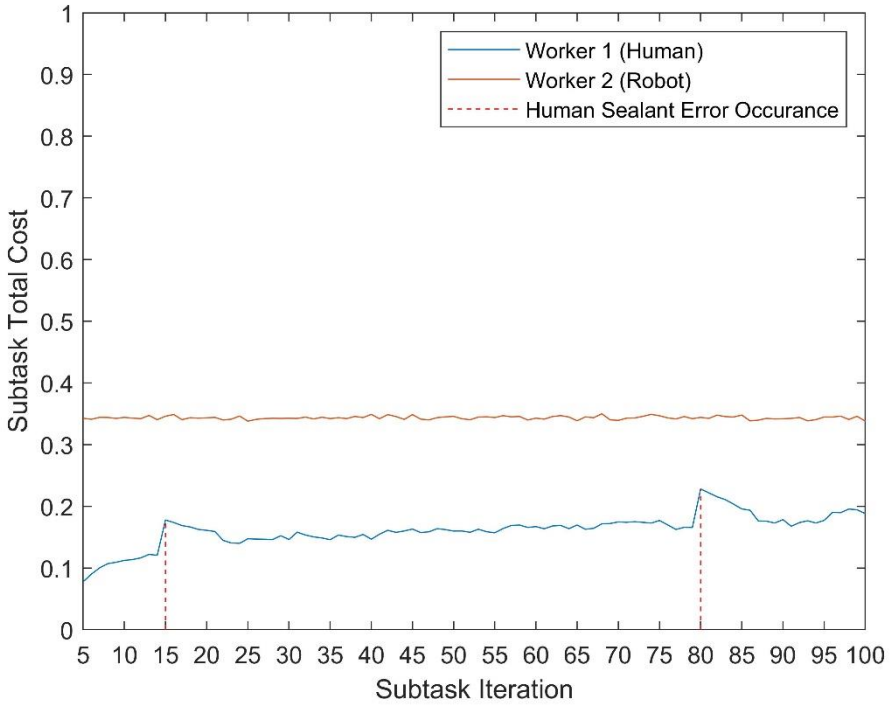

Figure 5.36: Plot of the total cost for the human and robot worker in the sealant application subtask with the infrequent occurrence of sealant trajectory errors for the human worker.

Next, it is necessary to analyse the group of scenarios where the same sealant application errors frequently occur, in these cases this is every 5 task

iterations between iterations 15 and 40 of the 100 simulated iterations of the subtask. For the first scenario consisting of frequent sealant coverage errors, Figure 5.37 shows the output of the discrete precision of sealant application variable and Figure 5.38 shows the output total cost of the complete cost function for the subtask. Figure 5.37 shows that the cost of the discrete precision of sealant application variable is not significantly affected for the first two error occurrences of 76% sealant area errors in task iterations 15 and 20. Further occurrences of sealant coverage errors quickly increases the cost of the variable with a peak cost of 0.7517 reached with the 26% sealant area error in task iteration 40. This peak output cost is approximately 2.4 times the cost of the 26% sealant area sealant area error and demonstrates the variable's severe reaction to repeated frequent errors in the sealant application subtask. After this final error, it takes 18 successfully completed iterations of the sealant application subtask for the human worker's cost to return to nominal levels. Figure 5.38 shows that the increase in the discrete precision of sealant application variable cost during these frequent sealant coverage errors has a significant effect on the total cost for the human worker to complete the sealant application subtask. For the occurrence of the 26% area error in task iteration 40, which represents the sixth error within 50 iterations of the subtask, it is shown that the cost for the human worker to complete the subtask at 0.3967 increases beyond that of the robot worker at 0.3490.

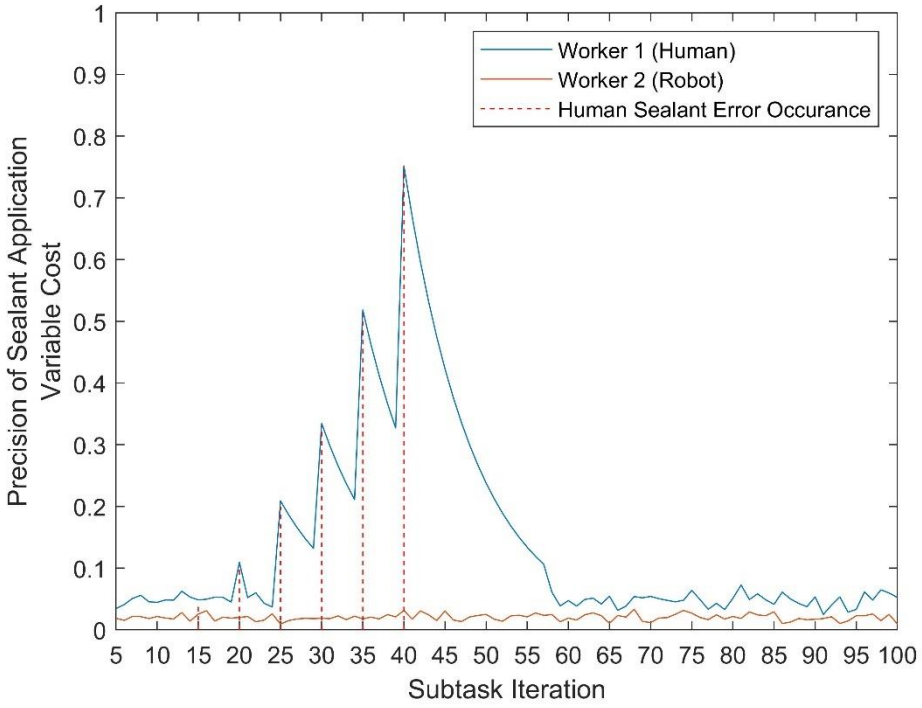

Figure 5.37: Plot of the discrete precision of sealant application variable cost for the human and robot worker in the sealant application subtask with the frequent occurrence of sealant coverage errors for the human worker.

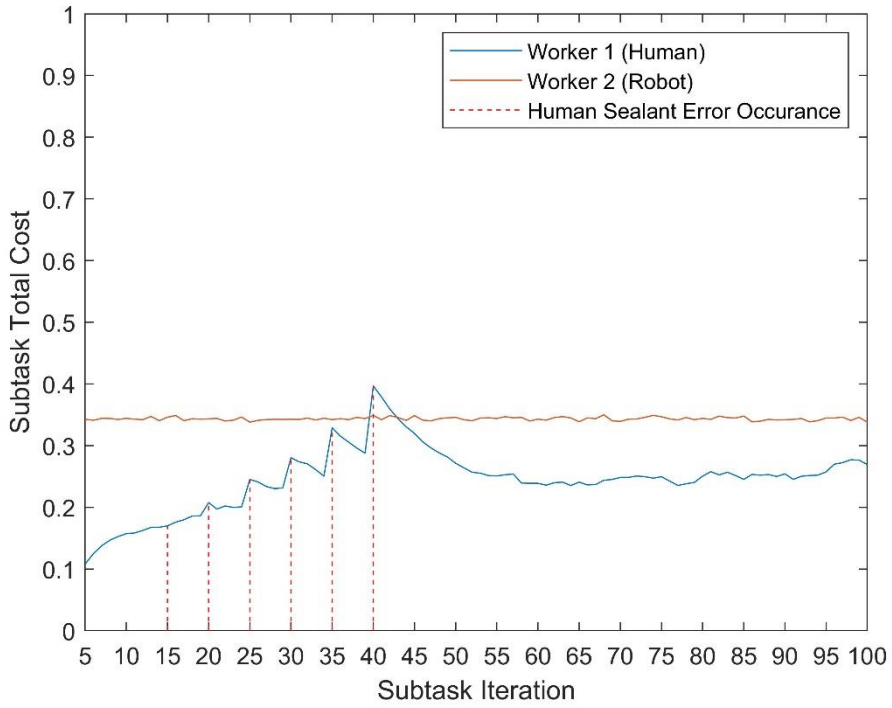

Figure 5.38: Plot of the total cost for the human and robot worker in the sealant application subtask with the frequent occurrence of sealant coverage errors for the human worker.

For the second scenario consisting of frequent sealant gap errors, Figure 5.39 shows the output of the discrete precision of sealant application variable and Figure 5.40 shows the output total cost of the complete cost function for the sealant application subtask. Figure 5.39 shows that the frequent occurrence of sealant gap errors causes a much quicker increase in cost for the discrete precision of sealant application variable in this scenario. The variable reaches a maximum possible cost of 1 with the 25% sealant length error in task iteration 35, a further occurrence of this error in iteration 40 again caused the variable to reach a cost of 1. After this final error, it takes 20 successfully completed iterations of the subtask for the human worker's cost to return to nominal levels. The total cost for the human worker to complete the subtask is again significantly affected by the increase in cost of the discrete precision of sealant application variable during this scenario as shown in Figure 5.40. Here it is shown that with the occurrence of the 50% length error in task iteration 30, which is only the fourth error within 50 iterations of the subtask, that the cost for the human worker to complete the subtask at 0.3842 increases beyond that of the robot worker at 0.3427. The total cost for the human worker to complete the subtask reaches a peak of 0.4588 with the occurrence of the 25% length error in task iteration 40, compared to a cost of 0.3490 for the robot worker.

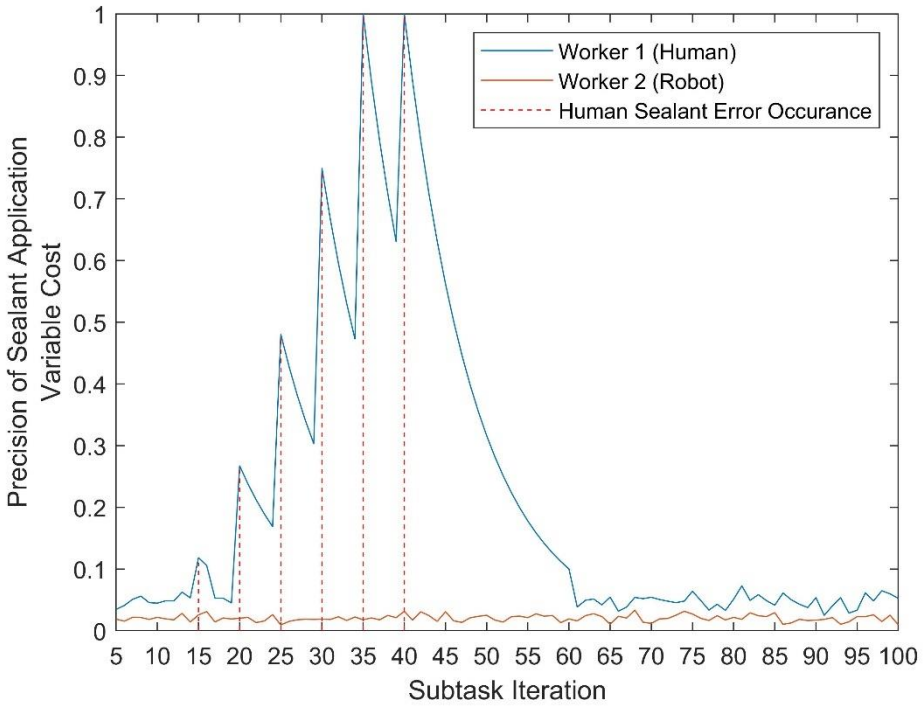

Figure 5.39: Plot of the discrete precision of sealant application variable cost for the human and robot worker in the sealant application subtask with the frequent occurrence of sealant gap errors for the human worker.

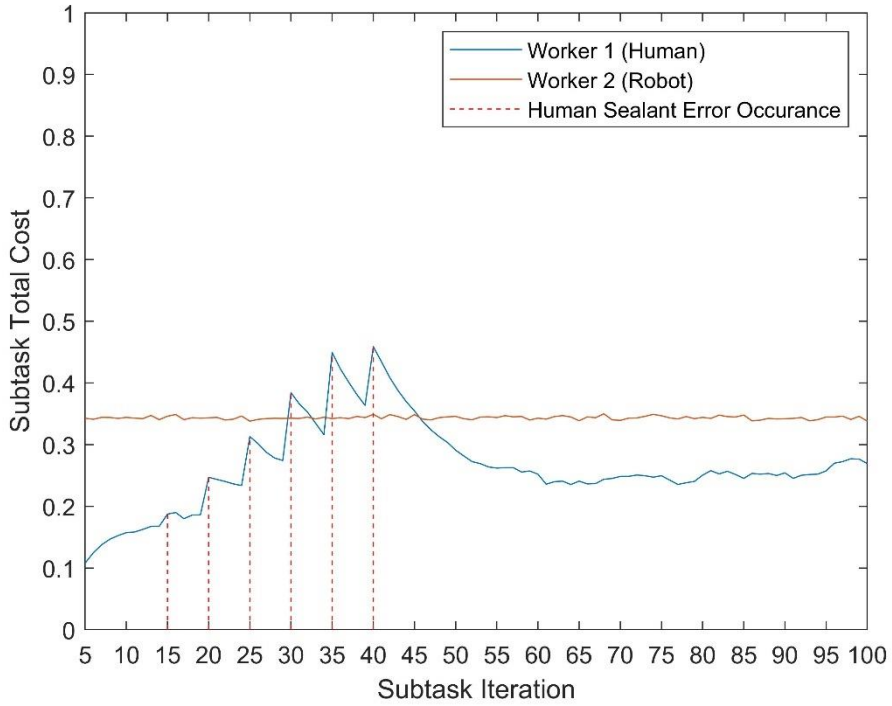

Figure 5.40: Plot of the total cost for the human and robot worker in the sealant application subtask with the frequent occurrence of sealant gap errors for the human worker.

For the third scenario consisting of frequent sealant trajectory errors, Figure 5.41 shows the output cost of the discrete precision of sealant

application variable and Figure 5.42 shows the output total cost of the complete cost function for the subtask. Figure 5.41 shows that the frequent occurrence of sealant trajectory errors causes the cost of the discrete precision of sealant application variable to increase far more rapidly than the other scenarios in this group. Here, the variable reaches its maximum cost of 1 in task iteration 30 with the 15° deviation error, occurrences of the 20° deviation error following this in task iterations 35 and 40 cause the variable to increase again to a cost of 1. After this final error, it again takes 20 successfully completed iterations of the subtask for the human worker's cost to return to nominal levels. Figure 5.42 shows that the increase in cost of the discrete precision of sealant application variable during these occurrences of frequent sealant trajectory errors has a significant effect on the total cost for the human worker to complete the sealant application subtask. Here it is shown that with the occurrence of the 15° deviation error in task iteration 25, which is only the third error within 50 iterations of the subtask, that the cost for the human worker to complete the subtask at 0.3891 increases beyond that of the robot worker at 0.3427. The total cost for the human worker to complete the subtask again reaches a peak of 0.4588 with the occurrence of the 20˚ trajectory error in task iteration 40, compared to a cost of 0.3490 for the robot worker.

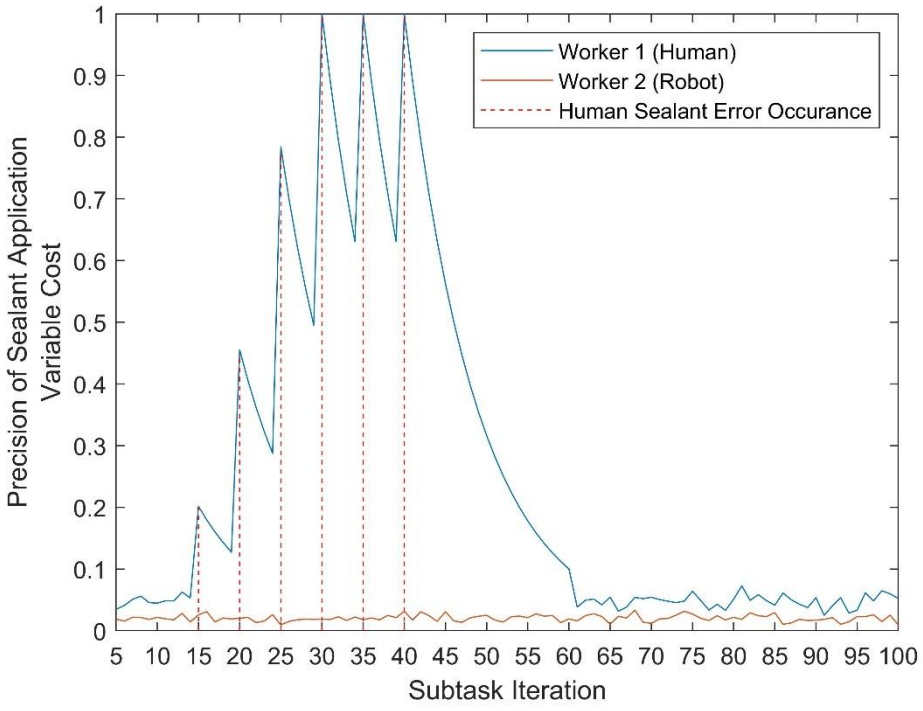

Figure 5.41: Plot of the discrete precision of sealant application variable cost for the human and robot worker in the sealant application subtask with the frequent occurrence of sealant trajectory errors for the human worker.

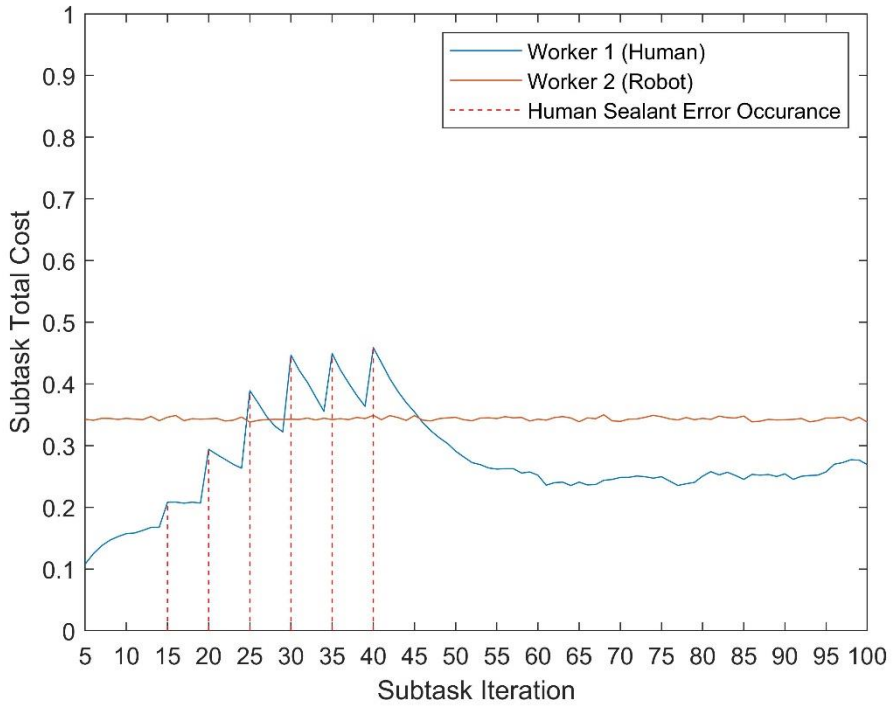

Figure 5.42: Plot of the total cost for the human and robot worker in the sealant application subtask with the frequent occurrence of sealant trajectory errors for the human worker.

Finally, it is necessary to analyse the scenario where error severity increases with each error occurrence for the human worker, here errors occur every 5 task iterations between iterations 15 and 30 of the 100 simulated iterations of the subtask. Figure 5.43 shows the output cost of the discrete precision of sealant application variable and Figure 5.44 shows the output total cost of the complete cost function for the subtask. Figure 5.43 shows that the first occurrence of a 26% area applied error only results in an insignificant increase in cost of the discrete precision of sealant application variable shows due to the low severity of the error. However, the cost of the discrete precision of sealant application variable rapidly increases with the increasing severity of errors and finally peaks at a cost of 0.99 with the final occurrence of the 20° trajectory deviation error in task iteration 30. In this scenario, it takes 20 successfully completed task iterations after this final error for the cost of the discrete precision of sealant application variable to return to nominal levels for the human worker. Analysing Figure 5.44 shows that the increase in cost of the discrete precision of sealant application variable in this scenario has a significant effect on the total cost for the human worker to complete the sealant application subtask. In this scenario, the occurrence of the 20° trajectory deviation error in task iteration 30, which is only the third error within 50 iterations of the subtask, causes the total cost for the human worker to complete the subtask to increase to 0.3403 slightly exceeding that of the robot worker at 0.3380. The total cost for the human worker to complete the subtask reaches a peak of 0.4443 with the occurrence of the 20° trajectory

deviation error in task iteration 30, compared to a cost of 0.3427 for the robot worker.

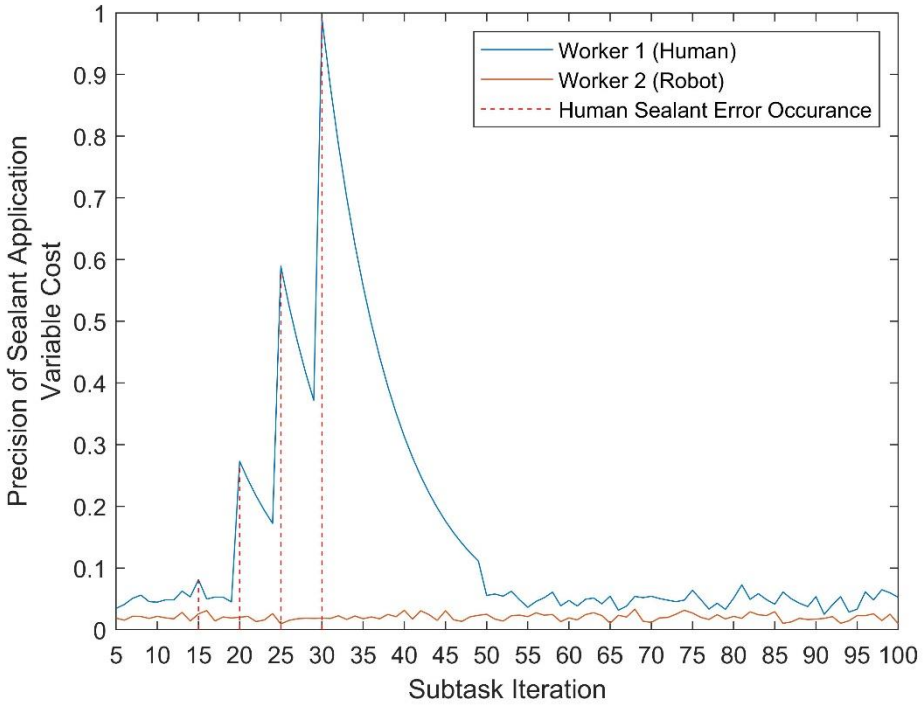

Figure 5.43: Plot of the discrete precision of sealant application variable cost for the human and robot worker in the sealant application subtask with the frequent occurrence of errors of increasing severity for the human worker.

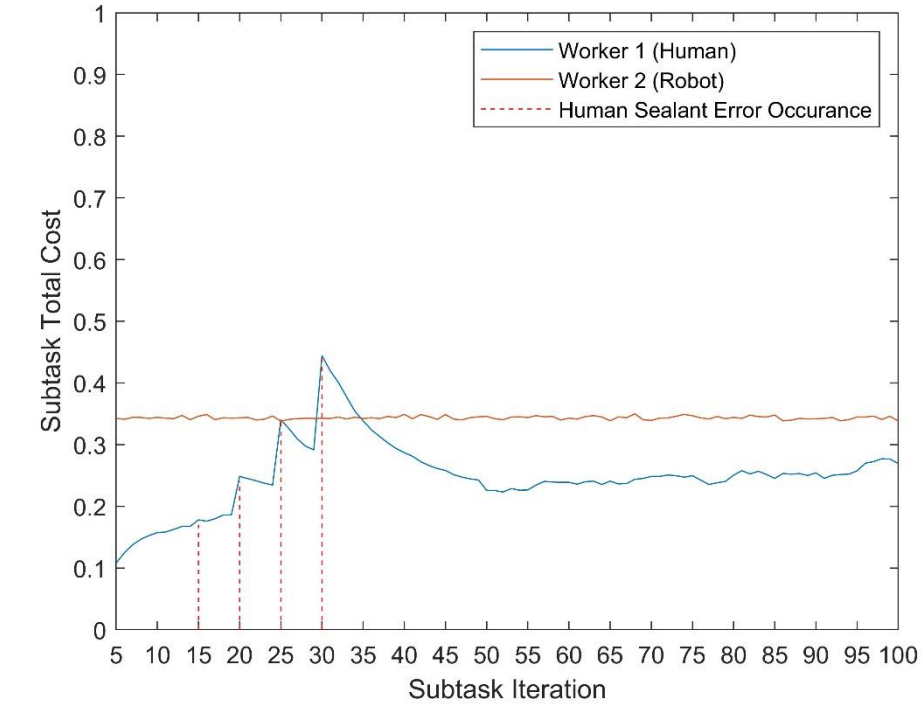

Figure 5.44: Plot of the total cost for the human and robot worker in the sealant application subtask with the frequent occurrence of errors of increasing severity for the human worker.

5.5. Chapter Summary

In this chapter, a methodology was proposed for a discrete variable for the dynamic cost functions proposed in this thesis to quantify the severity of instantaneous discrete events such as errors in the execution of manufacturing subtasks during a work shift. This discrete variable acts as a counterpart to the continuous variables proposed in Chapter 4 to ensure costs generated for workers also reflect their current capabilities in completing subtasks regarding factors such as quality of execution and accuracy. Given that such a variable must by its nature be highly specific to the type of task it is applied to, this methodology was demonstrated and tested through an example subtask of the application of a straight line of sealant.

Through this methodology the discrete variable principle involved first determining a series of parameters from a straight line of sealant through machine vision that can be used to determine its accuracy. Through this data it was first necessary to determine if an error had occurred in the sealant application and if so, determine the severity of the current error using this data and a predetermined hierarchy of discrete events. The variable must then determine its output cost based on the frequency and severity of previous errors, in addition to the current error, within a set number of task iterations. If an error has not occurred the variable must determine its output cost based on its output cost in previous iterations, reducing this with successfully completed task iterations then determining the output cost from data on the current sealant application.

This example discrete precision of sealant application variable was tested by first simulating several sealant application errors for a human and robot worker, to determine how the hierarchy of discrete events described in Section 5.3.5 or Eq. (5.1) described in Section 5.3.4 would grade them. It was shown in Section 5.4.2 that sealant application errors for the application of a straight line of sealant could be graded autonomously based on a still image of the sealant application on a workpiece located within a defined workspace. The simulated sealant applications for the human and robot worker successfully demonstrated the principle of the hierarchy of errors and Eq. (5.1) by determining if an error had occurred, grading the type of error and then grading the severity of the error within the hierarchy tier for that error.

Utilising these errors, it was next necessary to simulate the discrete variable and total cost across a work shift to determine the effect of discrete errors on the output costs of the sealant application subtask for the simulated human and robot workers. Work shifts were simulated with three groups of scenarios for possible error occurrences for the human worker. For the first group of scenarios consisting of isolated incidents of sealant application errors, it was shown that the discrete precision of sealant application variable was

tolerant of isolated errors from the human worker as these do not necessarily imply a change in worker capabilities. Here it was shown that isolated incidents of low severity errors did not have a noticeable effect on the discrete precision of sealant application variable and the total cost for the subtask. However, as the severity of the error increased a minor increase in the output cost of the variable and the total cost for the subtask was shown. Despite this increase for the most severe errors, the total cost for the human worker to complete the sealant application subtask was shown to remain significantly lower than that of the robot worker. This tolerance for isolated errors was necessary for the discrete variable as it was not expected that a human worker would be able to complete a task successfully 100% of the time over such long time periods and thus leniency was required for infrequent errors. It was considered appropriate that the total cost for the human worker to complete the subtask showed a minor increase for more severe errors as while this does not imply a change in capabilities, the severity of the error should be recognised.

For the second group of scenarios consisting of the frequent occurrence of a single type of sealant application error, it was shown that the discrete precision of sealant application variable reacted significantly to show the implied change in the human worker's capabilities. The behaviour of the discrete precision of sealant application variable in these scenarios was desirable as the frequent occurrence of errors resulted in the human worker's cost exceeding that of the robot worker with a lower number of error occurrences required to achieve this for more severe errors. This demonstrated the need for the discrete variable in the dynamic cost functions as a worker required speed and accuracy in completion of the subtask to be the optimal worker. The reaction of the variable given the severity of the errors was also desirable as it should take a larger number of low severity errors for the discrete precision of sealant application variable to reach its maximum value. This was because, as stated in Section 5.4.1, the frequent occurrence of low severity errors would likely indicate where a human worker is prioritising speed of completing the sealant application subtask too much over quality. This should not be considered as serious as frequent occurrence of higher severity errors which would likely indicate a change in the capabilities of the human worker.

In the third group of scenarios consisting of a single scenario where each error occurrence increased the severity of the error type, the output cost of the discrete precision of sealant application variable peaked at 0.99 with the fourth error occurrence which was very close to its maximum value. This behaviour was similar to that seen with the frequent occurrences of the most severe type of error despite the first two errors of this scenario being of a lower severity. It was again shown that the total cost for the human worker to complete the subtask exceeded that of the robot worker after three error occurrences by a very insignificant margin. However, the final error further

widened this cost increase over the robot worker. This behaviour of the discrete precision of sealant application variable in this scenario was again desired as frequent occurrences of errors of increasing severity implied that the human worker's capabilities were declining and reassigning the subtask they were completing is a justifiable reaction.

Following the investigation of continuous variables in Chapter 4 and discrete variables in Chapter 5, it is necessary to apply them both in full dynamic cost functions and apply them to subtasks of a full manufacturing task across a work shift. Utilising the costs generated for workers, it is necessary develop a task planning methodology to determine an optimum set of task assignments and task plan for a HR team.

6. Dynamic Task Planning of a Human-Robot Collaborative Manufacturing Task

6.1. Introduction

It is proposed to use a semi-online method of task planning for Human-Robot (HR) teams that utilises the advantages of online and offline planning methods as described in Chapter 3. It is also proposed to use real-world production indicators collected online during the execution of a manufacturing task, to quantify worker capabilities via the dynamic cost functions proposed in Section 3.6. These costs, based on worker capabilities, are then utilised by an offline task planner that can search for an optimal set of task assignments and task plan for a HR team. This task planner splits optimisation over two layers: Layer 1, which optimises task assignments, whilst Layer 2, searches for the optimal task plan for a given set of task assignments and acts as the objective function for Layer 1.

To find optimum task assignments quickly and efficiently an intelligent algorithm must be used to search the solution space. A metaheuristic search algorithm is chosen to do this as such algorithms prioritise exploration of the solution space in early iterations with exploitation of the best solutions being prioritised in later iterations of the search algorithm. As described in Section 3.3, the Discrete Gravitational Search Algorithm (DGSA) (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014) is chosen to find an optimal solution for worker task assignments in addition to the worker task plan. The DGSA is a discrete adaptation of the original Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-Pour and Saryazdi, 2009), a metaheuristic search algorithm that could only be used for continuous problems.

This chapter begins with a discussion of the methodology to generate a set of task assignments and task plans using the DGSA in Section 6.2. Section 6.2 begins by detailing a methodology for encoding a set of task assignments and a task plan for a HR team to enable the application of a search algorithm. Following this, the operating principle of the GSA proposed by (Rashedi, Nezamabadi-Pour and Saryazdi, 2009) is outlined in addition to the adaptations required to form the DGSA proposed by (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014). Using this definition of the DGSA, a dual-layer dynamic task planner is proposed to search for an optimal set of task assignments and task plan. To achieve this, the implementation of the DGSA is detailed for the first layer, which generates a set of task assignments, and the second layer which generates a task plan given a set of task assignments. This chapter next details a set of pre-execution constraints for this dynamic task planner in Section 6.3 to ensure workers are assigned subtasks when costs indicate they are significantly more suited to a subtask than the other workers. The dynamic task

planner is then tested in Section 6.4 through application in example cases to generate an initial set of task assignments and task plan using historic worker data. The first layer of the task planner is tested alone with a real manufacturing task with only two potential task plans. This manufacturing task is then modified with a simulated set of precedence constraints to increase the number of potential task plans, this allows testing of the second layer in addition to the complete dynamic task planner. These results are compared against the results from a brute force search of the solution space in terms of accuracy and speed followed by analysis and conclusions on its effectiveness.

6.2. Task Planning Using the Discrete Gravitational Search Algorithm

6.2.1. Task Plan and Task Assignments

A set of task assignments and a task plan must be generated to govern the optimal way for human and robot workers to complete a manufacturing task collaboratively. An optimal set of task assignments should ensure that the most appropriate worker is assigned to each assembly subtask by utilising the costs for each worker given by the cost functions described in Chapters 4 and 5. An optimal task plan should also ensure that assembly subtasks are ordered such that the cycle time for the HR team and idle times for its workers are minimised. Since task assignments and task plans are viewed from a high-level of abstraction in this research, and it is assumed that a lower level controller will execute subtasks for a robot as described in Chapter 3, task plans are formed around precedence relationships.

The task assignments and plans are both generated using precedence relationships and subtask specifications given as a part of an abstract assembly plan. This abstract assembly plan contains two core elements used by the dynamic task planner, the assembly plan and the assembly constraints. The assembly plan, *I,* is given as a list of *N* sequentially ordered subtasks such that

$$I = (1, \dots, N)$$

where each assembly subtask is a series of a primitive tasks that combine to carry out an element of the overall manufacturing task. A corresponding constraints list, *O*, represents the precedence relationships for each subtask of the assembly and is given by

$$O = (o_1, \dots, o_N),$$

where $o_i \subset I$ represents a list of subtasks that must be completed before the i^{th} subtask is executed. However, the list of subtasks given by o_i should only include the subtasks the i^{th} subtask is directly dependant on and not a complete list of all subtasks that must be complete before it is executed. When the i^{th}

subtask has no precedence relationships it is necessary to set $o_i = 0$ to represent that no subtask must be completed before the i^{th} subtask is executed. The assembly plan, *I*, and the constraints list, *O*, together represents a typical precedence graph of subtasks given for a manufacturing task such as that seen e.g. in Figure 6.1. As this shows, the nodes represent the subtasks of *I* such that

$$I = (1,2,3,4,5,6,7,8,9)$$

and the directed edges represent the execution constraints, o_j, on a subtask such that the corresponding constraints list, *O*, in this case is given by

$$O = (0,1,2,3,2,5,2,7,\{4,6,8\}).$$

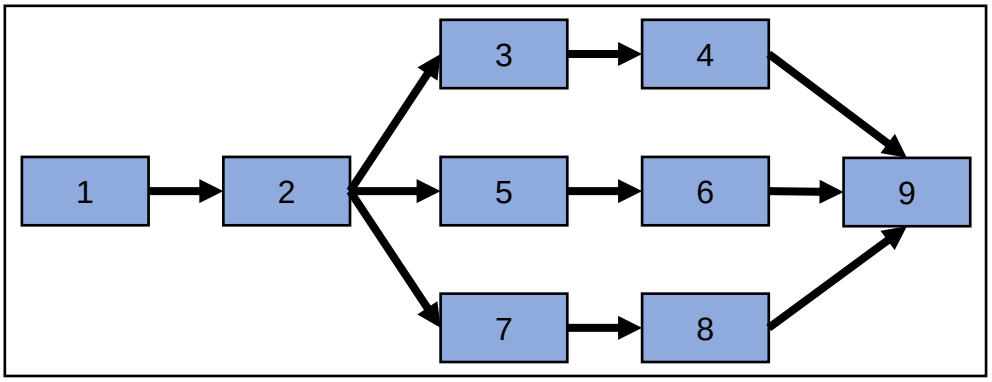

Figure 6.1: A representative set of assembly precedence constraints.

The assembly plan, *I*, and the constraints list, *O*, give the basis which allows the definition of sets of task assignments and task plans. When there are *W* available workers, numbering them sequentially allows a set of task assignments, *A*, to be given by

$$A = (\alpha_1, \dots, \alpha_N)$$

where $\alpha_i \in [1, W]$ gives the task assignment for the i^{th} subtask. A task plan, *P*, is defined as an ordering of the assembly plan, *I*, such that the task constraints, *O*, are satisfied and is given by

$$P = (p_1, \dots, p_N)$$

where $p_i \in I$ gives the i^{th} subtask to be completed in the task plan. The set of task assignments, *A*, and task plan, *P*, combine to give the instructions for a HR team to collaboratively complete a manufacturing task. Again, considering the example manufacturing task given in Figure 6.1, with $W = 2$ available workers where worker 1 is a human and worker 2 is a robot, a possible set of task assignments can be given by

$$A = (1,2,2,1,2,1,2,1,2)$$

with a corresponding task plan being given by

$$P = (1,2,7,8,3,4,5,6,9).$$

The task plan and set of task assignments is represented by the graph in Figure 6.2, where the colour of the nodes is used to visualise *A* with green and red nodes representing tasks assigned to human and robot workers, respectively. The path of directed edges of the graph in Figure 6.2 represents the corresponding task plan given by *P* which describes the order in which nodes are visited in a Hamiltonian path.

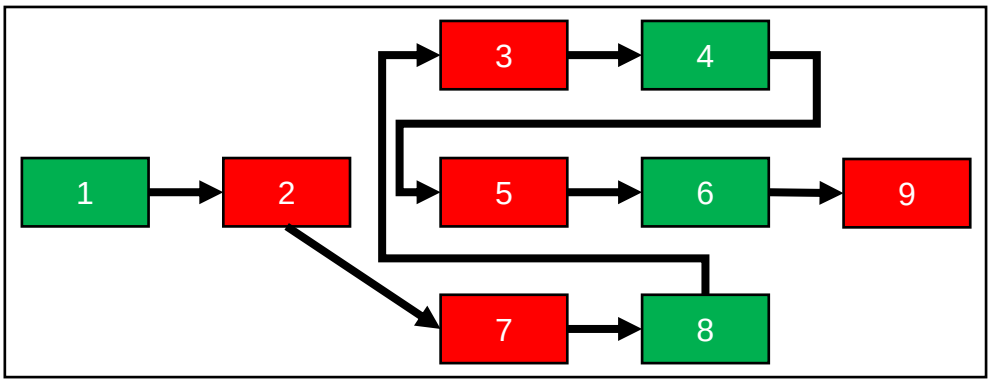

Figure 6.2: A representative combined task plan and set of task assignments, green nodes representing the human worker's assigned subtasks with red representing those of the robot.

Defining manufacturing task constraints along with sets of task assignments and plans in this way enables the definition of the solution space for finding the optimal way for a human and robot to complete a manufacturing task. Following this, it is necessary to apply a search algorithm to find the optimal solution. For this, a metaheuristic search algorithm is chosen as the speed of calculation is equally as important as the quality of the solution. To apply the chosen Discrete Gravitational Search Algorithm, it is first necessary to detail the operating principle of the original Gravitational Search Algorithm to define how the Discrete adaptation operates.

6.2.2. Gravitational Search Algorithm

The Gravitational Search Algorithm (GSA) is a metaheuristic agent-based search algorithm for solving optimisation problems, developed in (Rashedi, Nezamabadi-Pour and Saryazdi, 2009), inspired by the laws of gravity. The search algorithm was based on the concept of Newtonian gravity in that objects of mass accelerate towards each other via a gravitational force. Based on this concept, a population of *S* searcher agents are introduced into an *n* dimensional solution space where the position of the i^{th} agent, X_i, represents a potential solution to the optimisation problem such that

$$X_i = (x_i^1, \dots, x_i^d, \dots, x_i^n)$$

where x_i^d represents the position of the i^{th} agent in the d^{th} dimension of the solution space i.e. a variable of a potential solution. The motion of these searcher agents is simulated through the solution space using the Newtonian

laws of gravity to find an optimal solution to an optimisation problem. Each agent is given a mass based on the fitness of the solution, calculated using an objective function, with better performing agents being given a higher mass. The idea behind the method is that heavier agents will have a larger attraction radius and hence a greater intensity of attraction towards other agents with gravitational forces.

The search strategy operates by first injecting the *S* searcher agents into the solution space at random positions. The main phase of the search algorithm then progresses by moving the population of searcher agents towards the *K* best solutions, *Kbest*, of the current population where $K \in [0, S]$. This movement length in each dimension of the solution space is calculated using two components, a velocity representing an Independent Movement Length (IML) and an acceleration due to the gravitational forces from the *Kbest* solutions representing a Dependant Movement Length (DML). The IML for each agent is calculated by taking its previous velocity and multiplying it by a random real number in the range [0,1], with the initial velocity being randomly generated. This is independent as it only requires the agent's previous velocity (or movement length) and its current position. The DML is calculated by determining the acceleration of the agent due to the gravitational force on the agent by each of the *K* best agents. This is dependant as it requires knowledge of the positions of all the solutions in *Kbest* in addition to their masses. After each iteration of the search algorithm *K* and the gravitational constant, *G*, are reduced, the mass of each agent is re-evaluated, and the new *K* best solutions are found. The search algorithm will continue until *K* is reduced to one and an optimal solution is found. Reducing the gravitational constant, *G*, across iterations of the search algorithm results in reduction of the gravitational force between searcher agents. To achieve this, functions must be defined to reduce the value of *G* and *K* with each iteration, δ, of this main phase of the search algorithm. A possible method to achieve this is by defining a maximum number of iterations, λ, that the main phase of the search algorithm can execute. This method of exploring the solution space allows large exploration of the solution space in early iterations with exploitation of the best solutions found with more local searches in later iterations.

6.2.3. Discrete Gravitational Algorithm

The GSA was developed for use in optimisation problems with a continuous optimisation space (Rashedi, Nezamabadi-Pour and Saryazdi, 2009), however, Dowlatshahi et al (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014) developed the Discrete Gravitational Search Algorithm (DGSA) to apply the same process to optimisation problems with discrete solution spaces allowing its use in combinatorial optimisation problems. In comparison to the GSA, the DGSA takes the *Kbest* solutions from the lifetime of the execution of the DGSA, as a result of this the mass calculation for the

DGSA must include the combined populations of searcher agents, X_i, and *Kbest*. This is done since the mass of an agent is calculated via its fitness relative to the fitness of other agents thus the masses of the *Kbest* solutions must be recalculated with each iteration of the search algorithm. The DGSA also replaces the IML and DML with an Independent Movement Operator (IMO) and Dependant Movement Operator (DMO), respectively. The IMO and DMO allow movement through the solution space as operators since the DGSA has a discrete solution space as opposed to the continuous solution space of the GSA which requires the use of the continuous IML and DML. The DGSA carries out the independent movement for each agent via a modified local search algorithm, the modification stops the local search algorithm if the *IML* is equal to the number of iterations of the search algorithm carried out or when no better neighbouring solution exists.

To apply the DML in a discrete solution space, (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014) outlined the concept of a Neighbourhood space within the solution space to define movement around the solution space. This Neighbourhood space is defined by an undirected graph where nodes correspond to potential solutions and edges correspond to movements in the Neighbourhood space. Such movements between two neighbour potential solutions are carried out by a small move operator, φ, with a small move, m. An example of this corresponding to the Travelling Salesman Problem would be a swap operator represented by φ between an arbitrary pair of swappable cities represented by m. In this example the swap operator would be applied to a potential solution to swap the positions of two cities within the potential solution sequence to form a neighbour solution. (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014) use the concept of path relinking to define a path between two solutions in the solution space through applications of a small move operator. Using this concept, it is then possible to define a partial path between two solutions by additionally using a movement length that gives the number of applications of a small move operator to generate the path.

(Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014) applied the DMO by using the concept of generating partial paths between solutions to move a searcher agent towards each of the *Kbest* solutions. The movement length towards each of the *Kbest* solutions was calculated by the acceleration of the searcher agent towards them due to the gravitational force on the agent given their masses. This movement of the searcher agent along partial paths to the members of *Kbest* is applied individually and results in each movement starting from a different position. Due to this, (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014) defined an order of precedence to determine the priority with which these movements were applied. This precedence was defined by the fitness of the members of *Kbest*, the highest priority was given to target

solutions with the highest fitness value and the lowest priority is given to the target solutions with the smallest fitness value.

These adaptations to the Gravitational Search Algorithm produced the Discrete Gravitational Search Algorithm. As stated in Section 3.3, this search algorithm was chosen as it is appropriate for use with combinatorial optimisation problems such as the Traveling Salesman Problem. If the task planning problem is presented in a directed graph where nodes represent subtasks and edges represent assembly constraints, then it can also be considered as a path planning problem. This allows the task planning problem to be considered as finding a Hamiltonian path from the first subtask where all subtasks must be visited once in an optimal path. The DGSA can also be applied to the task assignment problem which is instead presented as a permutation problem.

In this research it is proposed to utilise dual layers of the Discrete Gravitational Search Algorithm (DGSA) to find an optimal set of task assignments in addition to an optimal task plan within one optimisation process. In this process Layer 1 optimises the task assignments with Layer 2 operating as the fitness function for Layer 1 by finding the optimal task plan and cost for a given set of task assignments. This ensures that a balance is achieved between optimising the selection of the best worker for each subtask with optimising the manufacturing task as a whole. The operating principle for the dynamic task planner is shown in Figure 6.3, demonstrating how the two layers of the DGSA interact with each other to find the optimum set of task assignments and task plan. Here the abstract assembly plan and cost function generator, as described in Chapter 3, act as input to the dynamic task planner, providing the costs for workers and manufacturing task specifications required by the dynamic task planner. DGSA Layer 1 uses the potential task assignments for the manufacturing task as the solution space, the fitness function for this layer then passes the potential task assignment input into it to DGSA Layer 2. DGSA Layer 2 uses potential task plans as its solution space, the fitness function uses worker costs along with the resulting idle times for a potential task plan and set of task assignments to find the optimal task plan for this potential set of task assignments. DGSA Layer 2 then returns the optimal task plan to the fitness function for DGSA Layer 1, with the associated cost being used as the output of the fitness function in DGSA Layer 1. DGSA Layer 1 then outputs the optimal task plan and set of task assignments so that they can be executed by the HR team. Using this methodology, it is next necessary to detail the application of the DGSA to each of the layers of this process.

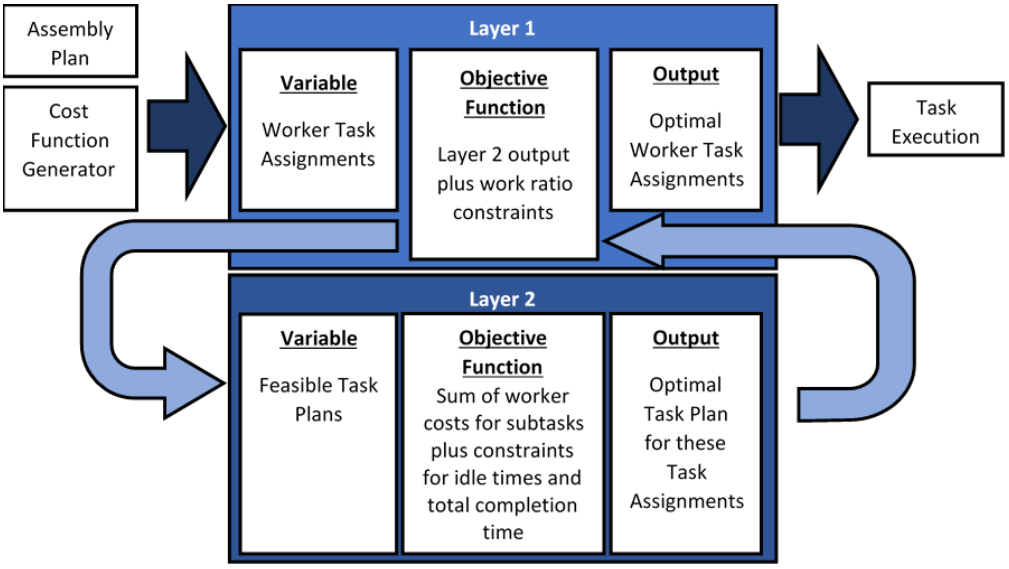

Figure 6.3: The framework for the dynamic task planner, detailing the interaction between Layer 1 and Layer 2, necessary to find the optimal set of task assignments and task plan for a HR team.

6.2.4. DGSA Layer 1 – Task Assignment Determination

The first layer of the DGSA is used to optimise task assignments by formulating it as a permutation problem. To define the problem in terms of the DGSA, it is necessary to first define the solution space and composition of the searcher agents. Using the Assembly Plan, I, definition given in Section 6.2.1 containing *N* subtasks, results in an *N* dimensional solution space. The defined format of task assignments, A_i, also defines the format of the searcher agent's positions, where the position of the i^{th} searcher agent, X_i, is given by a potential task assignment, A_i and is defined as

$$A_i = \left(\alpha_i^1, \dots, \alpha_i^d, \dots, \alpha_i^N\right)$$

where $\alpha_i^d \in \{1, \dots, W\}$ represents the task assignment for the d^{th} subtask when there are *W* available workers.

The fitness function representing the objective function of the optimisation problem for this layer operates by running Layer 2 of the dynamic task planner as described later in Section 6.2.5. DGSA Layer 2 outputs an optimal task plan, *P*, in the format described in Section 6.2.1 and the task plan cost for the full manufacturing task. This optimal task plan is stored alongside the task assignment and cost for later output if they are associated with the optimal set of task assignments. From this, the fitness value, F_i, for the i^{th} searcher agent at position X_i in the search algorithm is given by the optimal cost for the task assignment output from DGSA Layer 2. A special case is defined when 75% or greater of the subtasks in a set of task assignments, A_i, are assigned to a single worker as it is assumed that both workers must always be used. In such cases, it is undesirable for DGSA Layer 2 to be utilised since this set of task assignments should not be used. The fitness value for the i^{th} searcher

agent at position X_i of the search algorithm, F_i, in such situations is then given as $F_i = 2N$, twice the sum of the maximum cost for a worker to complete each subtask, and the optimal task plan for the task assignment is not given. This is done to ensure that such an undesirable set of task assignments is not used and that DGSA Layer 2 is not executed wasting processing time.

This layer of the DGSA follows the standard format set out in Section 6.2.3 using the standard IMO and DMO to move agents around the solution space, however, to apply this a small move operator, φ, and small move *m* must be defined. Since this is formulated as a permutation problem, the small move operator for DGSA Layer 1, φ_1, is defined as changing a single task assignment in the position of the *i*th agent A_i. The small move, m_1, is then defined as from the *j*th worker to the *k*th worker in dimension $d \in \{1, .., N\}$ such that $\alpha_i^d = j$ and $k \in \{1, \dots, W\}$ but $j \neq k$. An example application of φ_1 with move m_1 in dimension $d = 1$ for a search space where N = 3 with $W = 2$ available workers, on a searcher agent $A_i = (1,2,1)$ would be

$$A_i = A_i \, \varphi_1 \, m_1 = (2,2,1).$$

Operating DGSA Layer 1 with these inputs and this small move operator outputs the optimum task assignment for the HR team alongside the optimum task plan and cost generated from DGSA Layer 2. In addition, it is necessary to set the number of searcher agents, *S*, and the stopping conditions for the main phase of the search algorithm. However, these variables are dependent on the optimisation problem and its solution space. It is also necessary to set functions to reduce the size of the variable K to govern the size of $Kbest$ and to govern the size of the gravitational constant, $G(\delta)$, for each in iteration δ of the main phase of the search algorithm. However, these functions are dependent on the stopping conditions for the main phase of the search algorithm. Since these inputs and functions are problem dependant, they are defined for the example cases given in Section 6.4.

6.2.5. DGSA Layer 2 – Task Plan Determination

The second layer of the DGSA was used to find the optimal task plan for a given set of task assignments from DGSA Layer 1, using worker costs and production indicators. Unlike DGSA Layer 1 this is considered as a combinatorial problem and must find the optimal task plan by manipulating the order of subtask execution within potential task plans. To define the problem in terms of the DGSA, it is again necessary to define the solution space and composition of the searcher agents. Using the task plan, *P*, defined in Section 6.2.1 the *i*th task plan is defined as

$$P_i = (p_i^1, \dots, p_i^d, \dots, p_i^N)$$

where $p_i^d \in I$ is a subtask in the assembly plan, *I*, such that the task constraints given by the constraints list, *O*, representing the precedence relationships for each subtask of the assembly given by

$$O = (o_1, \dots, o_d, \dots, o_N),$$

where o_d are the execution constraints for subtask p_i^d with $O_d \subset I$ are satisfied. Within this definition, there is the possibility of subassemblies, *Q*, occurring within the task plan consisting of several sequentially numbered subtasks, such that $Q \subset I$. The *i*th subassembly, Q_i, is defined by

$$Q_i = (q_i^1, q_i^2, \dots, q_i^k) = (j, j+1, \dots, j+k-1)$$

where q_i^k is the *k*th subtask for a subassembly containing *k* subtasks beginning with the *j*th subtask. Subassemblies must also have the unique corresponding constraints list, L_i, such that $L_i \subset O$ given by

$$L_i = (o_j, o_{j+1}, \dots, o_{j+k-1}) = (o_j, j, \dots, j+k-2)$$

for a subassembly containing *k* subtasks beginning with the *j*th subtask. The task plans operate on the principle that these subassemblies are completed in totality before new tasks or subassemblies are started as it is undesirable to have multiple subassemblies in partial states of completion with work constantly switching between subtasks. An exception to this is that the first subtask of a subassembly which can be completed in parallel to the final subtask of the previous subassembly, provided that task precedence relations are not broken. This exception is acceptable as one worker will remain idle whilst the other worker is completing the final subtask of a subassembly which would leave them available to start the next subassembly.

The presence of these subassemblies poses difficulty for the application of the DGSA as a small move operator would have to swap subassemblies consisting of various numbers of subtasks to reorder *P*, increasing the complexity of its operation. Combining this increased complexity due to subassemblies with the complex restrictions on the ordering of subtasks due to the overall constraints list, *O*, would produce a highly complex small move operator, φ, and small move, *m*. In contrast to the example application of the DGSA to the Travelling Salesman Problem given in (Nikolakis et al., 2018), the small move operator cannot be a simple swap operator due to the complex restrictions of *O*. Due to these factors it is required to encode the Task Plan, P_i, in a simpler format which allows a simpler small move operator, φ, to move agents through the solution space.

An encoding method is developed to simplify potential task plans into a new simplified task plan, B_i, and define a corresponding list of swappable task plan elements, *U*, such that a small move operator, φ, can be applied to B_i to make a small move, *m*, defined using *U*. Due to the dependency of this method

on the task plan it is applied to, it is described by utilising an example case. In this example it is necessary to consider a task plan, P_i, given by

$$P_i = (p_i^1, \dots, p_i^{14}) = (1,2,3,6,7,8,4,5,9,12,10,11,13,14)$$

with corresponding constraints list to its assembly plan, *I*, being given by

$$O = (o_1, \dots, o_{14}) = (0,1,2,1,4,1,6,1,\{3,5,7,8\},9,10,9,9,\{10,11,12,13\}).$$

To generate the simplified task plan, B_i, all the subassemblies, Q_j, are identified in the task plan which in this case are

$$Q_1 = (q_1^1, q_1^2) = (2,3),$$

$$Q_2 = (q_2^1, q_2^2) = (4,5),$$

$$Q_3 = (q_3^1, q_3^2) = (6,7)$$

and

$$Q_4 = (q_4^1, q_4^2) = (10,11).$$

All subtasks of these subassemblies except the lead element, q_j^1, are then removed from the Task Plan, P_i, to give the simplified task plan, B_i, such that

$$B_i = (b_i^1, \dots, b_i^{10}) = (1,2,6,8,4,9,12,10,13,14).$$

The removed subtasks are stored in a missing subtasks list, *V*, such that

$$V = (3,5,7,11)$$

and their execution constraints, o_j, are given by the corresponding constraint list, *Z*, such that

$$Z = (2,4,6,10)$$

to reform the full task plan, P_i, when required. Note that *V* and *Z* will be the same for all potential task plans and only need to be calculated once in the execution of DGSA Layer 2.

To generate *U*, first subtasks in the potential task plan which are dependent on the completion of more than one subtask in *O* are identified as static subtasks that cannot be swapped and given an identifier of 0, in the example case above such subtasks would be subtasks 9 and 14. In opposition to this, swappable task plan elements are identified as subtasks where $o_i = o_j$ for the i^{th} and j^{th} subtasks thus they can be swapped and still form a viable simplified task plan. These swappable task plan elements are given a common identifier sequentially as a positive integer in *U*, respectively, for example for $j, k, s, v \in B_i$ with $j < k < s < v$ and $o_j = o_k \neq o_s = o_v$ then j and k are represented by 1 in *U* whereas *s* and *v* are represented by 2. A special case is defined by a subtask acting as the single starting point for a potential task plan, this would be a case where the constraints list for the first subtask $o_1 = 0$ and

is unique. This special case would also result in the subtask being defined as a static subtask and being given an identifier of 0 in *U*. Combining these rules forms the steps required to find *U* which is given as

$$U = (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}) = (0,1,1,1,1,0,2,2,2,0)$$

in the example case discussed above.

This encoding method allows the definition of the solution space and searcher agents used by DGSA Layer 2 for the task planning problem. Using the simplified task plan definition, B_i, if this consists of j subtasks and subassemblies, results in a j dimensional solution space. This definition also provides the format of the searcher agent's positions, where the position of the i^{th} searcher agent, X_i, is given by a potential simplified task plan, B_i, and is defined as

$$B_i = \left(b_i^1, \dots, b_i^d, \dots, b_i^j\right)$$

where b_i^d represents the d^{th} subtask or subassembly to be completed in the task plan.

The fitness function for DGSA layer 2 must produce the overall cost for a given simplified task plan and task assignment to enable the best task plan to be found in DGSA Layer 2 and the best task assignment to be found when costs are output to DGSA Layer 1. The fitness function for DGSA Layer 2 must include the cost for each worker to complete individual subtasks as given by the dynamic cost functions given in Section 3.6 in addition to cost penalties for the quality of the overall task plan in relation to worker idle times and overall completion times for the task plan. This is done to ensure a balance between assigning tasks to the best possible workers and efficient use of workers in the overall task plan to optimise production rates.

To calculate the fitness value, F_i, of a task plan, first it is necessary to decode the simplified task plan, B_i, to the full task plan, P_i, as the details of all subtasks are required to calculate the cost. In addition to this, the task assignment, A_i, input into DGSA Layer 2 from the fitness function of DGSA Layer 1 as given by

$$A_i = (\alpha_i^1, \dots, \alpha_i^d, \dots, \alpha_i^N)$$

is also required to calculate the cost. The fitness, F_i, of the i^{th} searcher agent of DGSA Layer 2 is then given by

$$F_i = \sum_{k=1}^{N} (C_{k,\alpha_k} + J_{k,\alpha_k}) + M$$

where C_{k,α_k} is the cost for worker α_k assigned to complete the k^{th} subtask given by the dynamic cost functions in Section 3.6, J_{k,α_k} is the idle cost and M is the

total completion time cost. The idle cost is used to ensure that idle times for workers are not excessive, ensuring the balance between selecting the best worker for each task and efficient worker utilisation is kept. Since it is impractical to have no idle times, a scale is defined to quantify the severity of idle times and calculate the idle cost J_{k,α_k}. To provide such a scale a maximum acceptable idle time, θ, is defined for workers to allow the definition of the idle costs as

$$J_{k,\alpha_k} = \frac{\rho_{k,\alpha_k}}{\theta}$$

where ρ_{k,α_k} is the idle time for worker α_k whilst waiting to start the k^{th} subtask. As with task information given in the abstract assembly plan described in Chapter 3, it is assumed that the maximum acceptable idle time, θ, would be defined by the manufacturer implementing the system. The total completion time cost, *M*, is used here to ensure that the total completion time, *l*, of a set of task assignments is not excessive to again ensure a balance between selecting the best worker for each task and the optimisation of the task as a whole. To define such a cost, a scale is required to quantify if the total completion time for the workers to complete a manufacturing task is acceptable. To provide such a scale it is necessary to use the manufacturer's desired work element times, H_j, used by the completion variable for the j^{th} subtask allowing the definition of the total completion time cost as

$$M = \frac{l}{\sum_{k=1}^{N} H_k}.$$

This layer of DGSA also follows the standard format set out in Section 6.2.3 and uses the standard IMO and DMO to move agents around the solution space. However, in order to apply the IMO and DMO, a task plan, P_i, must be encoded in its simplified form, B_i. As this problem is a combinatorial problem the small move operator for DGSA layer 2, φ_2, is defined as swapping the location of two subtasks in B_i with each other. The small move, *m*, is then defined as between a pair of subtasks *j* and *k* in dimensions *s* and *v* such that $b_i^s = j$ and $b_i^v = k$ with the constraint that $u_s = u_v$.

Operating DGSA Layer 2 in this way outputs the optimum task plan and cost for the HR team given the input task assignments used in DGSA Layer 1. To operate DGSA Layer 2 it is also necessary to set the number of searcher agents, *S*, and the stopping conditions for the main phase of the search algorithm, however, these variables are dependent on the optimisation problem and its solution space. It is also necessary to set functions to reduce the size of the variable K to govern the size of $Kbest$ and the size of the gravitational constant, $G(\delta)$, for each in iteration δ of the main phase of the search algorithm. However, these functions are again dependent on the stopping conditions for the main phase of the search algorithm. Since these inputs and

functions are problem dependant, as with DGSA Layer 1, they are defined for the example cases given in Section 6.4.

6.3. Task Planner Pre-Execution Constraints

To find an optimum task plan efficiently and quickly it is desirable to first minimise the solution space of possible solutions by eliminating undesirable solutions. It is proposed to use pre-execution constraints for the task planner to pre-allocate tasks for workers based on the worker costs given by the dynamic cost functions in Section 3.6. This only applies to cases where there is a significant difference in cost between workers and ensures assignment of a subtask to the best worker in the generated set of task assignments and task plan. The additional benefit of task pre-assignment is that for each task pre-assigned before the task planner is executed, the solution space of possible task assignments is reduced by a factor of *W* available workers to complete the subtask.

To enable such a pre-execution constraint, it is first necessary to define what would be considered a significant difference in cost between two workers, which requires the definition of a threshold cost, η, by the manufacturer implementing the system. This threshold cost allows distinction between workers and should be set sufficiently high to ensure that the better worker must receive the task assignment if it would be inappropriate to assign another worker to the task. To ensure the task assignment is maintained by the task planner it is necessary to define a list of "locked in" task assignments, Ψ, of the same size as the task assignments, A_i, given by

$$\Psi = [\psi_1, \dots, \psi_N]$$

where ψ_j is a "locked in" identifier for the j^{th} subtask. Given the threshold cost each "locked in" identifier, ψ_j, is given by

$$\psi_j = \begin{cases} 1 \text{ if } C_{j,s} - C_{j,k} \geq \eta \text{ for all } s \in \{1, \dots, W \mid s \neq k\} \\ 0 \text{ otherwise} \end{cases}$$

where *k* is the proposed best worker and *s* represents another potential available worker. This "locked in" list instructs the task planner that a task assignment can be changed if $\psi_j = 0$ and cannot be changed if $\psi_j = 1$. To define which worker is assigned a "locked in" task assignment, a corresponding set of base task assignments, Λ, is defined as

$$\Lambda = [\alpha_1, \dots, \alpha_N]$$

where

$$\alpha_j = \begin{cases} k \text{ if } \psi_j = 1 \text{ for worker } k \\ 0 \text{ otherwise} \end{cases},$$

here a task assigned to worker 0 remains a placeholder for assignment by the dynamic task planner.

6.4. Dynamic Task Planner Testing and Results: Generating Initial Set of Task Assignments and Task Plan

6.4.1. Generating a Single Set of Task Assignments and Task Plan From Historic Data

To test the dynamic task planner proposed in this chapter, an example manufacturing task is utilised given by Nikolakis et al. (Nikolakis et al., 2018). This is chosen as it is a real manufacturing task for the assembly of a turbocharger and has been studied with relation to task planning in HR teams. The task information given for this assembly task is also at a suitably high level of abstraction to be utilised by the dynamic task planner and provides details on task duration times for a human and robot worker. In this manufacturing task, (Nikolakis et al., 2018) allow some subtasks to be completed by humans and robots in direct collaboration. However, this option will not be used within this research with the subtask only being assigned to either the human or robot worker.

The dynamic task planner is first tested for a static set of completion times to represent the case of generating an initial set of task assignments and task plan, where real time production data is not yet available. Such a set of task assignments and task plan would be executed by the HR team until enough data is gathered to begin to assess the current capability and performance of the workers via the full dynamic cost functions, following which the task assignments and task plan would be re-evaluated. To apply the dynamic task planner, it is first necessary to define how costs should be generated for the workers, this will require pared down cost functions that are capable of being used with historic data only. These cost functions will only use the completion time variable in this case. However, to ensure weightings remain consistent across the work shift the fatigue variable and precision of sealant application variable will be included but remain at zero. The precision of sealant application variable is used in these cost functions since the details of quality assurance methods for subtasks that would be used to form discrete variables were not provided in (Nikolakis et al., 2018) for the turbocharger assembly task. Utilising the precision of sealant application variable thus allows the effect of discrete errors on task replanning utilising the dynamic task planner to be tested.

To define the costs given by the completion variable used in this initial set of task assignments and task plan, it is first necessary to define the historic initial completion times for the human worker as defined in Section 4.2. These historic initial completion times are defined using the task duration times for the human worker given by (Nikolakis et al., 2018) and are given in Table 6.1.

These are used as these task duration times represent the expected performance of the human worker thus are suitable to represent their historic mean initial completion times. It is also necessary to define the completion times for robot workers, these are again defined by the task duration times for the robot workers given by (Nikolakis et al., 2018) as seen in Table 6.1. These are used as the expected completion times for a robot worker should vary minimally across a work shift under normal operation thus the expected completion times provide a good measure of the robot's performance for the initial costs. Finally, to calculate the completion cost function variable for both human and robot workers it is also necessary to define the manufacturer's desired work element time for each assembly subtask. It is assumed that the manufacturer's desired work element time would be slightly lower than that of the best worker historically, thus these work element times for this task are set at 90% of the completion time for the fastest worker and are given in Table 6.1.

Table 6.1: Table of human historic initial completion times, initial robot completion times and manufacturer's desired work element times for each subtask for the example manufacturing task.

Subtask Number	Human Historic Initial Completion Time (Seconds)	Initial Robot Completion Time (Seconds)	Manufacturer's Desired Work Element Time (Seconds)
1	6	12	5.4
2	4.7	9	4.23
3	16.6	18	14.94
4	8.1	10	7.29
5	3.9	7	3.51
6	1.7	3	1.53
7	7.9	11	7.11
8	18	21	16.2
9	19	23	17.1
10	5.3	9	4.77
11	2.4	4	2.16
12	1.7	5	1.53
13	2.7	6	2.43
14	3.3	8	2.97
15	8.5	10	7.65
16	8.8	15	7.92
17	13.5	25	12.15
18	9.5	12	8.55
19	1.9	4	1.71
20	2.4	3	2.16
21	7.2	8	6.48
22	12.3	15	11.07
23	17.7	20	15.93
24	2.6	4	2.34

Following this it is necessary to define the weightings utilised for the cost functions for each worker in each subtask of the overall manufacturing task. The subtasks in the example manufacturing task are grouped into three general categories of subtask, screwing, pick & place and sensing, weightings are defined for each of these and applied to all subtasks in that category. For each of these categories of subtasks, it is necessary to break down the subtask into its primitive elements to determine their weightings as described in Section 3.5. Starting with the screwing task, this is considered a primitive task and thus cost function variables that affect the task are weighted equally, with those that do not being given a weighting of zero. For this subtask, the completion and fatigue variables are used for the human worker, and only the completion variable is used for the robot worker. An error checking variable is not used in this case as errors that could occur, e.g. wrong bolt used or not correctly tightened, would not be able to be detected with autonomous visual checks such as those in Section 5.3. Due to this the analogous discrete precision of sealant application variable is given a weighting of zero. Second for the pick and place tasks, the subtask is broken down into checking the specification for where to place components, picking and placing the components then verifying the operation has been completed correctly. For this subtask with the human worker, the completion variable would only affect the physical movement of components with the fatigue and error detection variables instead affecting all primitives of the subtask, both cognitive and physical. For the robot worker the completion variable again only affects the physical primitive of the task with the error detection variable again affecting all primitives of the subtask. Finally, for the sensing subtasks (Nikolakis et al., 2018) do not detail what this entailed so it is assumed that such tasks should be used as the verification of correct completion of assemblies or visual inspection of parts. In this case it is determined that for both workers a sensing task should not be dependent on completion times as successful completion would be more important if the sensing subtask is vital to the manufacturing task. Acknowledging this, it is determined that the fatigue and error detection cost function variables should have an equal effect on all primitives of the subtask for the human worker with only the error detection variable affecting the primitives for the robot worker. The effects of cost function variables detailed combined with the weighting schema detailed in Section 3.5. results in the cost function weightings for each worker for each subtask in Table 6.2.

Table 6.2: Cost function variable weightings for human and robot workers.

		Cost Function Variable		
Manufacturing Task	**Worker**	**Completion**	**Fatigue**	**Precision of Sealant Application**
Screwing	Human	1/2	1/2	0
	Robot	1	0	0
Pick & Place	Human	1/7	3/7	3/7
	Robot	1/4	0	3/4
Sensing	Human	0	1/2	1/2
	Robot	0	0	1

Table 6.3: Type of manufacturing task definition for each subtask and their precedence constraints.

Subtask Number	Type of Manufacturing Task	Precedence Constraints
1	Pick & Place	0
2	Pick & Place	0
3	Pick & Place	2
4	Sensing	{1,3}
5	Pick & Place	4
6	Pick & Place	5
7	Pick & Place	6
8	Screwing	7
9	Screwing	8
10	Pick & Place	9
11	Pick & Place	10
12	Pick & Place	11
13	Pick & Place	12
14	Pick & Place	13
15	Pick & Place	14
16	Screwing	15
17	Screwing	16
18	Pick & Place	17
19	Pick & Place	18
20	Pick & Place	19
21	Screwing	3
22	Screwing	21
23	Pick & Place	22
24	Sensing	23

In addition to the costs relating to the subtasks, it is also necessary to define how to calculate costs relating to the idle times of workers. To achieve this, it is necessary to define the maximum acceptable idle time, θ, for each

subtask of the overall manufacturing task. Here θ is set as 18 seconds as the manufacturer's desired work element times for subtasks in this manufacturing task, given in Table 6.1, lie in the range of 1.53 to 17.1 seconds. It is assumed here that the maximum acceptable idle time should be no longer than the length of the longest subtask.

6.4.2. Simple Manufacturing Task

The dynamic task planner also requires the details of the overall manufacturing task and the precedence relationships between the subtasks to be executed. These details would be passed from the abstract assembly plan as given in Table 6.3 and illustrated in Figure 6.4 and are based on the precedence relationships for the assembly subtasks given in Table 6.3. In this example assembly task, the abstract assembly plan included the assembly plan, *I*, given by

$$I = (1, \dots, 24)$$

and the subtask execution constraints, *O*, given by

$$O = (0,0,2,\{1,3\},4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,3,21,22,23).$$

Due to these constraints and using the definition of a task plan in addition to the definition of DGSA Layer 2, there are only two possible task plans for this manufacturing task. This small number of potential task plans for each set of task assignments means that it is inappropriate to use DGSA Layer 2 in this case and instead the function of DGSA Layer 2 is executed by a brute force check of the potential solutions.

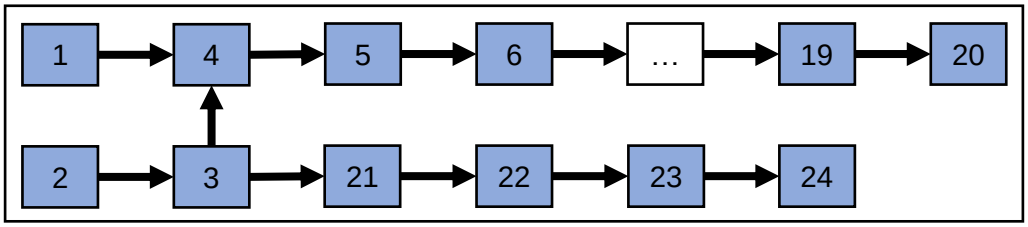

Figure 6.4: The precedence relationships for the turbocharger assembly task given in (Nikolakis et al., 2018).

Finally, to execute the dynamic task planner it is necessary to set the parameters of DGSA Layer 1 required for its operation. These parameters include the number of searcher agents, *S*, the stopping condition for the main phase of the search algorithm and the functions to reduce *K* and the gravitational constant, *G*, as the search algorithm progresses. In this example manufacturing task, having two available workers results in 16,777,216 possible sets of task assignments giving quite a large solution space. Given the size of the solution space it is chosen to test DGSA Layer 1 with various settings of these parameters to assess the impact on execution time and the accuracy

of solutions. The number of searcher agents, *S*, is the first of these parameters to be varied as this has an impact on the ability of DGSA to search the solution space. Next, the stopping condition for the main phase of DGSA Layer 1 is determined by setting a maximum number of iterations, λ, that the main phase of the search algorithm can execute and is the other setting to be varied. This stopping condition allows the use of linear decreasing functions to reduce the gravitational constant, $G(\delta)$, and the number of best solutions, *K*, over the lifetime of the execution of DGSA Layer 1. This also allows maximum and minimum values to be defined for the gravitational constant, $G(\delta)$, and the number of best solutions, *K*, with the progression between the two being defined by the current iteration number, δ, of the main phase of DGSA. Utilising this method over the lifetime of the main phase of DGSA layer 1, *K* is reduced from the number of searcher agents *S* to 1 via

$$K = S - \left(\frac{\delta}{\lambda} \times (S - 1)\right). \tag{6.1}$$

In addition to this $G(\delta)$ is reduced over the lifetime of the main phase of the DGSA Layer 1 from $G_{initial}$ to G_{end} via

$$G(\delta) = G_{initial} - \left(\frac{\delta}{\lambda} \times (G_{initial} - G_{end})\right). \tag{6.2}$$

The setting of $G_{initial}$ and G_{end} is problem specific and is set as $G_{initial} = 0.4$ and $G_{end} = 0.2$ for DGSA layer 1 as determined by trial and error.

Utilising these parameters, the dynamic task planner is tested for various numbers of searcher agents, *S*, and maximum number of iterations, λ, in DGSA Layer 1. This begins with a baseline, using *S* = 5 searcher agents over a maximum number of λ = 25 iterations of the main phase of the search algorithm. From this baseline, *S* is increased to 200 in steps of 5 and λ is increased to 50 or 100 to form a grid search to determine the effect on execution time and accuracy of solutions by altering these settings. For each of these settings, the dynamic task planner is run 30 times for the simple manufacturing task to determine accuracy of the solution and execution time compared to the optimum set of task assignments and task plan generated using a brute force search of all possible task plans for every possible set of task assignments. This brute force search was achieved by generating all possible sets of task assignments and task plans that satisfy the result of the task pre-execution constraints then applying the objective function to each of them to generate a data set. The set of task assignments and task plans with the minimum cost was found by searching this data set using built in Minimum functions of the MATLAB software package. Here, the dynamic task planner and the brute force search of the solution space are both implemented in the MATLAB software package. Analysis of the results also enables the determination of the optimum number of searcher agents and maximum number of iterations to be executed used by DGSA Layer 1 for this task planning problem.

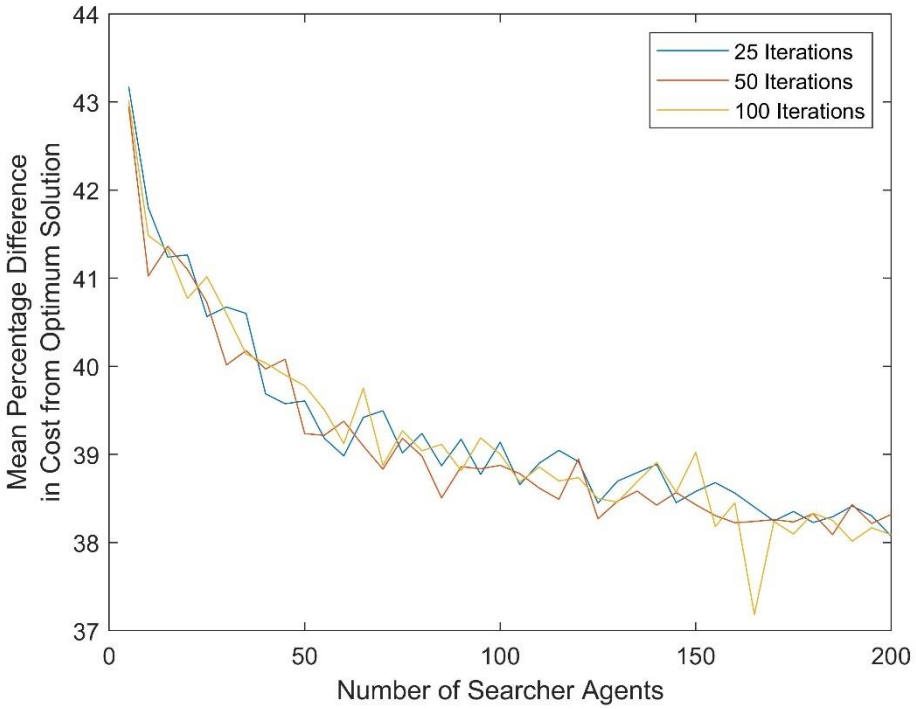

Figure 6.5: Plot of the mean percentage cost difference between the solution found by the task planner using only DGSA Layer 1 and the optimum solution found using brute force, with varying number of searcher agents and maximum number of iterations used by the main phase of DGSA Layer 1.

First, the accuracy of the task planner using only DGSA Layer 1 is evaluated by determining the percentage difference in cost between the solutions found by the task planner and the optimum solution found through brute force, as shown in Figure 6.5. It is first important to note that a brute force search of all potential solutions in the solution space finds a single optimum set of task assignments and task plan which has a cost of 8.76. It is shown that using the lowest number of searcher agents with $S = 5$, that the cost of the found set of task assignments and task plan is much higher than the optimal solution. The worst accuracy is achieved when using a maximum number of $\lambda = 25$ iterations in the main phase of DGSA Layer 1 at a mean difference in cost of 43.17% from the optimal solution, followed by 43.02% difference with $\lambda = 100$ and 42.95% difference with $\lambda = 50$. This behaviour is shown with a low number of searcher agents since, if a searcher agent gets trapped in a local minimum in an IMO phase of DGSA Layer 1 it will attract others towards it in the next DMO phase causing them to be trapped as well. The effect of this is that the DMO will not cause agents to navigate the solution space if all searcher agents are trapped at the same position and the IMO phase cannot move solutions out of the local minimum. In such situations, the maximum number of iterations of the main phase of DGSA Layer 1 used will not affect performance and the variation in accuracy seen here is likely due to the position of the randomly generated initial positions of agents within the solution space.

Figure 6.5 shows that increasing the maximum number iterations in the main phase of DGSA Layer 1 has an inconsistent effect on the accuracy of the solution. However, increasing the number of search agents used by DGSA Layer 1 consistently improves the accuracy of solutions. Considering the number of searcher agents used when a maximum of λ = 25 iterations in the main phase of DGSA Layer 1, increasing the number of searcher agents to *S* = 50 resulted in a mean percentage cost difference of 39.61% from the optimum solution. Increasing the number of searcher agents to *S* = 100 results in a diminishing return in the improvement in the accuracy of the search algorithm with a mean percentage cost difference of 38.66%. Finally, increasing the number of searcher agents to *S* = 200 results in a greater diminished return in the improvement in accuracy with a mean percentage cost difference of 38.07%.

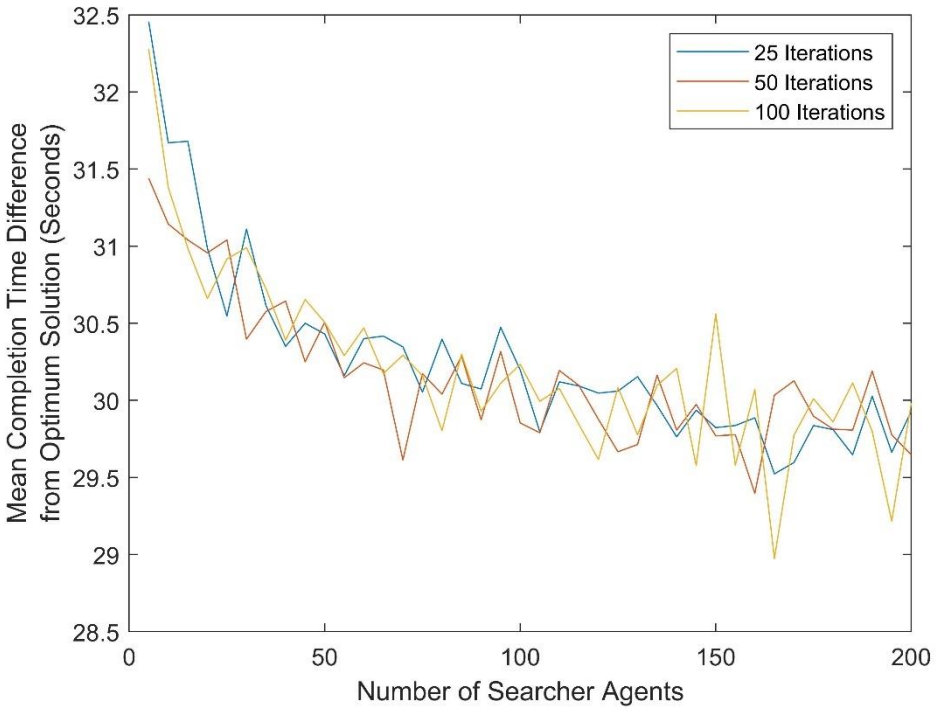

Figure 6.6: Plot of the mean difference in completion time of the simple manufacturing task between the solution found by the task planner using only DGSA Layer 1 and the optimum solution found using brute force, with varying number of searcher agents and maximum number of iterations used by the main phase of DGSA Layer 1.

To test the accuracy of the task planner using only DGSA Layer 1, it is also necessary to compare the difference in total completion time between the sets of task assignments and task plans the task planner finds and the optimum solution found through brute force, given in Figure 6.6. It is first important to note that a brute force search of all possible solutions in the solution space found an optimum set of task assignments and task plan with a total completion time of 157.6 seconds. Figure 6.6 shows that with *S* = 5 searcher agents the worst mean increase in total completion time of 32.46 seconds from the optimal solution occurs when DGSA Layer 1 uses a maximum number of *λ*

= 25 iterations in its main phase. Similarly, a mean increase in total completion time of 31.44 seconds is shown with λ = 50, however, the mean increase in completion time is 32.28 seconds with λ = 100. Given the optimal completion time of 157.6 seconds, this means at worst the completion times are approximately 19.9% to 20.6% larger than that of the optimal solution. However, the task plan is not solely optimising the completion time of the simple manufacturing task. This shows that the mean completion time difference of sets of task assignments and task plans found by the task planner from the optimum solution when S = 5 and λ = 50 is smaller than those found when S = 5 and λ = 100.

Figure 6.6 again shows that increasing the number of searcher agents used by DGSA Layer 1 has a greater effect on the accuracy of solutions than increasing the maximum number of iterations used in its main phase. Again, considering the number of searcher agents used when a maximum of λ = 25 iterations in the main phase of DGSA Layer 1, using S = 50 searcher agents increases the accuracy of solutions with a mean increase in completion time of 30.43 seconds from the optimum solution. Increasing the number of searcher agents to S = 100 again results in a diminishing return in the improvement of the accuracy of the search algorithm with a mean increase in completion time of 30.2 seconds from the optimum solution. Increasing the number of searcher agents to S = 200 again results in a further diminishing return in the improvement in the accuracy of the search algorithm with a mean increase in completion time of 29.93 seconds from the optimum solution.

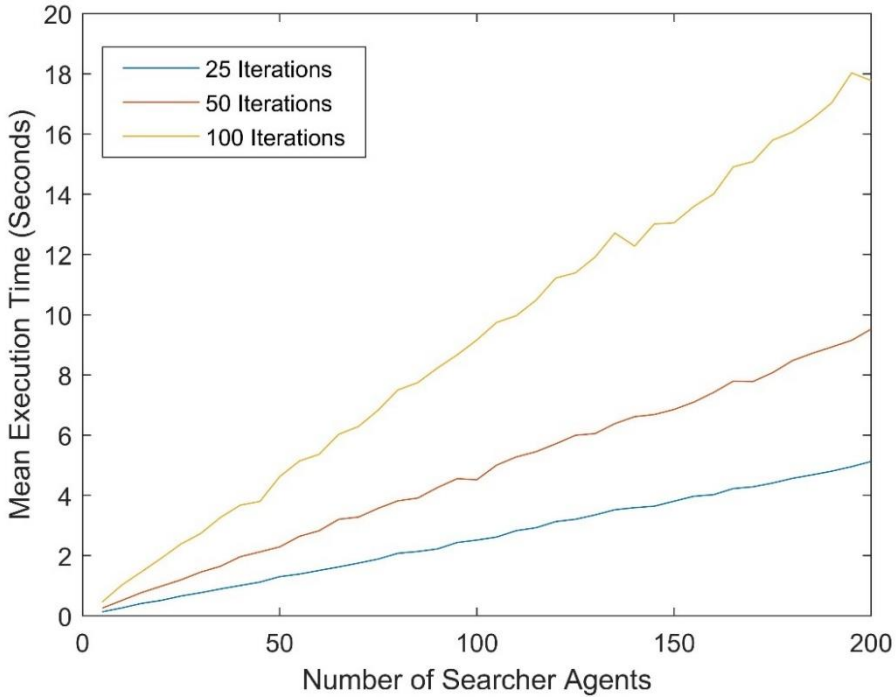

Figure 6.7: Plot of mean execution time of DGSA Layer 1 with varying number of searcher agents and maximum number of iterations used by the main phase of DGSA Layer 1. It is important to note that the

brute force search of the solution space had an execution time of 687.4 seconds, but this is far beyond the region shown in this plot.

Following analysis of the task planner's accuracy for the simple manufacturing task, it is necessary to analyse the mean execution time compared with a brute force search of each possible task plan for all possible task assignments to determine its utility. It is first important to note that using the brute force search of the solution space, the optimal set of task assignments and task plan are found in an execution time of 687.4 seconds. Figure 6.7 shows that the mean execution time of the task planner displays a linearly increasing trend with an increase in the number of searcher agents, S, in the main phase of DGSA Layer 1. However, the maximum number of iterations it uses, λ, significantly affects the rate of this growth. Comparing the mean execution time for the dynamic task planner utilising only DGSA Layer 1 with its accuracy, it is shown that increasing the number of searcher agents that DGSA Layer 1 uses has a greater effect on the accuracy of the task planner than increasing the maximum number of iterations it uses in its main phase. This implies that priority should be given to increasing the number of searcher agents DGSA Layer 1 uses rather than the maximum number of iterations it uses in its main phase.

For the dynamic task planner in this task planning problem, the mean execution time with a maximum of $\lambda = 25$ iterations in the main phase of DGSA Layer 1 increases from 0.134 seconds with $S = 5$ searcher agents to 1.3 seconds with $S = 50$ searcher agents and 2.52 seconds with $S = 100$ searcher agents. When DGSA Layer 1 uses $S = 200$ searcher agents it has a mean execution time of 5.13 seconds, meaning that with the tested settings the dynamic task planner takes 0.0195% to 0.746% of the time of the brute force method. Despite this small execution time with the highest settings tested, the optimal settings for DGSA Layer 1 in this example case are to use between $S = 50$ and $S = 100$ searcher agents with a maximum of $\lambda = 25$ iterations in the main phase of the search algorithm. This is determined since the greatest increase in accuracy is seen when increasing the number of searcher agents to $S = 50$ with diminishing returns in accuracy seen when more than $S = 100$ searcher agents are used. Additionally, utilising a more complex objective function in the search algorithm, such as implementing DGSA layer 2, will massively increase this execution time so it is undesirable to use more searcher agents given the minimal increase in accuracy of the task planner.

6.4.3. Complex Manufacturing Task

Since the assembly task given by Nikolakis et al. (Nikolakis et al., 2018) provides minimal opportunities to reorder subtasks to improve the efficiency of the HR team it is required to develop a simulated more complex test case to demonstrate the capabilities of DGSA Layer 2. It is chosen to use the same example assembly task as given in (Dowlatshahi, Nezamabadi-Pour and

Mashinchi, 2014) but with a different set of precedence relations for subtasks of the overall manufacturing task. These precedence constraints, given in Table 6.4 and illustrated in Figure 6.8, produce a manufacturing task with a higher number of possible subtask orderings using the definition of a task plan in addition to the definition of DGSA Layer 2.

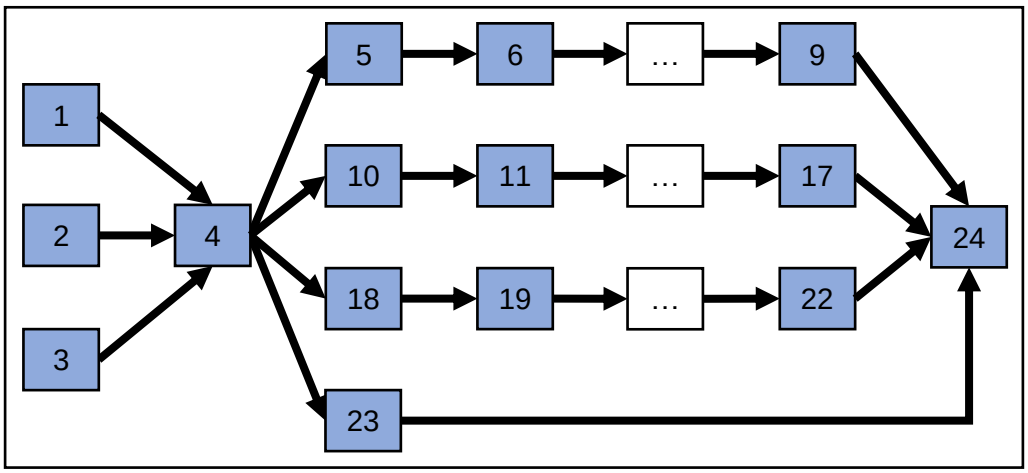

Figure 6.8: The precedence relationships for the simulated more complex assembly task.

Table 6.4: New subtask precedence constraints for simulated more complex assembly task.

Subtask Number	Precedence Constraints
1	0
2	0
3	0
4	{1,2,3}
5	4
6	5
7	6
8	7
9	8
10	4
11	10
12	11
13	12
14	13
15	14
16	15
17	16
18	4
19	18
20	19
21	20
22	21
23	4
24	{9,17,22,23}

This assembly plan allows the formation of multiple subassemblies, Q_i, as discussed in Section 6.2.5. which in this case are given by

$$Q_1 = [5,6,7,8,9],$$

$$Q_2 = [10,11,12,13,14,15,16,17]$$

and

$$Q_3 = [18,19,20,21,22].$$

These subassemblies allow the formation of a potential simplified task plan given by

$$B_i = [1,2,3,4,5,10,18,23,24]$$

and illustrated in Figure 6.9, along with its corresponding list of swappable task plan elements given by

$$U = [1,1,1,0,2,2,2,2,0],$$

missing elements list given by

$$V = [6,7,8,9,11,12,13,14,15,16,17,19,20,21,22]$$

and missing elements execution constraints given by

$$Z = [5,6,7,8,10,11,12,13,14,15,16,18,19,20,21].$$

Using the list of swappable task plan elements, there are 3 swappable subtasks in group 1 and 4 swappable subtasks and subassemblies in group 2 giving a total of 144 possible task plans for each set of task assignments.

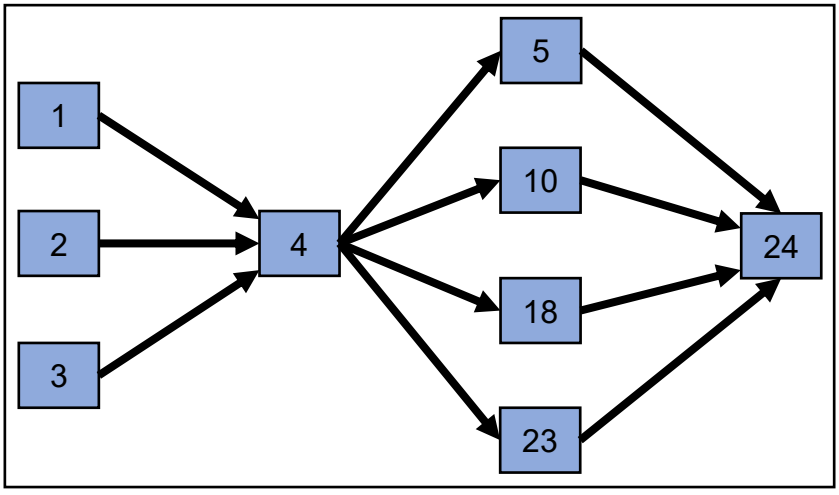

Figure 6.9: A potential simplified task plan for the assembly plan defined using the precedence relationships in Figure 6.8.

To execute the dynamic task planner, it is also necessary to set the parameters of both DGSA Layer 1 and DGSA Layer 2 required for its operation. These parameters are the same as those for DGSA Layer 1, however, now they must be set individually for DGSA Layer 1 and DGSA Layer 2. As with the

previous manufacturing task, having two available workers results in 16,777,216 possible task assignments, however, for each of these there are 144 possible task plans. This results in 2,415,919,104 possible solutions to the combined task assignment and task planning problem. In this simulated manufacturing task DGSA Layer 1 will utilise the optimal parameters determined for the real manufacturing task in Section 6.4.2. However, for the execution of DGSA Layer 2 the parameters of the number of searcher agents, *S*, and the maximum number of iterations, λ, executed by the main phase of the search algorithm are again varied to determine the effect on execution time and accuracy of solution. DGSA layer 2 also uses linear decreasing functions to reduce the gravitational constant, $G(\delta)$, and the number of best solutions, *K*, over the lifetime of the DGSA execution. Utilising this method, *K* is again reduced from the number of searcher agents *S* to 1 via Eq. (6.1) and $G(\delta)$ is reduced from $G_{initial}$ to G_{end} via Eq. (6.2). The setting of $G_{initial}$ and G_{end} are again problem specific and are given by $G_{initial} = 0.4$ and $G_{end} = 0.2$ for DGSA Layer 2 in this simulated case.

Utilising these parameters, it is first necessary to determine the optimum number of searcher agents, *S*, and maximum number of iterations, λ, for the execution of the main phase of DGSA layer 2. As with the dynamic task planner utilizing only DGSA Layer 1, this is necessary to determine the settings for DGSA Layer 2 to optimise its accuracy and execution time. This begins with a baseline using S = 2 searcher agents over a maximum number of λ = 4 iterations of the main phase of the search algorithm. From this baseline S is increased to 10 in steps of 1 and λ is increased to 10 in steps of 2 to form a grid search to determine the effect on execution time and accuracy of solutions. Such small values are used for these parameters in comparison to those used by DGSA Layer 1 due to the much smaller solution space of the task ordering problem to generate a task plan. For each of these settings, DGSA Layer 2 is run 30 times for six potential sets of task assignments of the complex manufacturing task, given in Table 6.5, to determine the accuracy of the solution and execution time compared to the optimum task plans generated by a brute force search of all possible task plans for a potential set of task assignments. This brute force search was achieved by generating all possible task plans for a potential set of task assignments then applying the objective function to each of them to generate a data set. The task plans with the minimum cost were found by searching this data set using built in Minimum functions of the MATLAB software package. Here, DGSA Layer 2 and the brute force search of the solution space are again both implemented in the MATLAB software package.

Table 6.5: Potential sets of task assignments used to test DGSA Layer 2.

Name of Set of Task Assignments	Task Assignments
potential task assignments 1	[1,1,2,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,1]
potential task assignments 2	[1,1,2,1,1,1,2,1,2,2,2,1,2,2,1,1,2,2,2,2,1,1,1,1]
potential task assignments 3	[1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2]
potential task assignments 4	[1,1,2,2,1,1,2,1,1,1,1,1,1,1,2,1,1,2,1,1,1,1,2,2]
potential task assignments 5	[1,2,1,2,2,2,1,2,1,2,1,2,1,2,1,2,1,1,1,2,1,2,1,2]
potential task assignments 6	[1,1,2,2,1,1,2,1,1,2,1,1,1,1,2,1,1,1,1,1,1,1,2,2]

Table 6.6: A table showing a comparison of the highest mean cost of task plans found by DGSA Layer 2 in comparison to those of the optimal task plans found via a brute force search of the solution space for the potential sets of task assignments given in Table 6.5. Additionally, the number of searcher agents required to reduce mean percentage cost difference to zero regardless of the maximum number of iterations, λ, used in the main phase of the search algorithm are given.

Potential Sets of Task Assignments	Cost of Optimum Solutions Found by Brute Force Search	Highest Mean Cost of Solutions Found with DGSA Layer 2 and Settings Used	Highest Mean Percentage Cost Increase of Solutions Found with DGSA Layer 2	Number of Searcher Agents, *S*, Required to Reduce Mean Percentage Cost Difference to Zero
1	15.67	15.88 $S = 2, \lambda = 10$	1.37%	7
2	11.98	11.98 $S = 2, \lambda = 4$	0%	2
3	13.48	13.62 $S = 2, \lambda = 6$	1.04%	5
4	9.58	9.77 $S = 2, \lambda = 6$	2.02%	10
5	14.07	14.16 $S = 2, \lambda = 4$ and $\lambda = 10$	0.67%	6
6	9.04	9.04 $S = 2, \lambda = 4$	0%	2

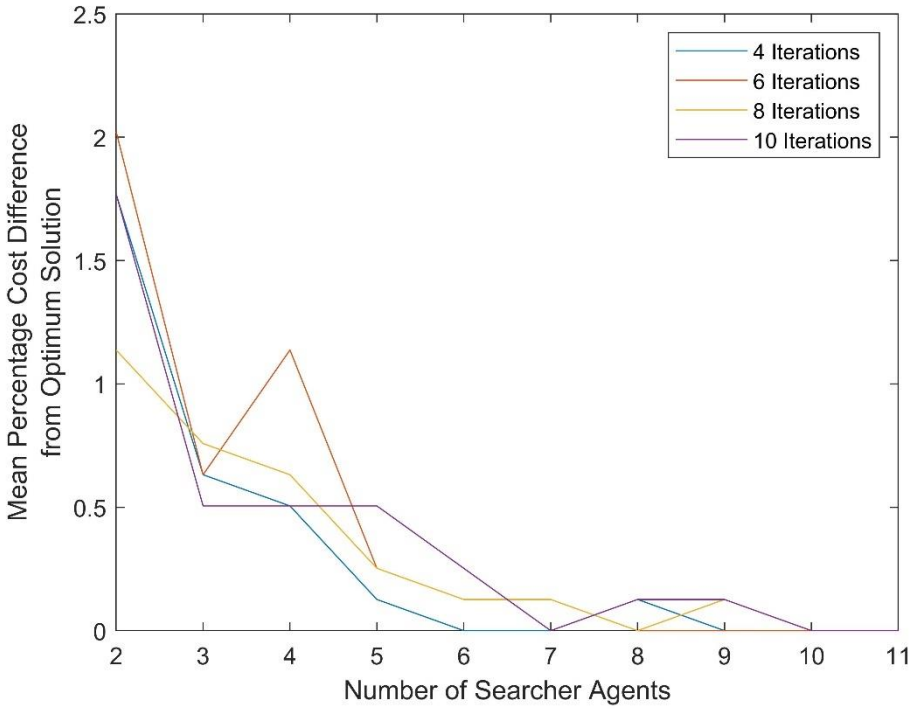

Figure 6.10: Plot of the mean percentage cost difference between the task plans found for potential task assignments 4 by DGSA Layer 2 and the optimum task plans found using brute force, with varying number of searcher agents and maximum number of iterations used by the main phase of DGSA Layer 2.

First, the accuracy of DGSA Layer 2 is evaluated by comparing the cost of the solutions DGSA Layer 2 finds for the potential sets of task assignments in Table 6.5, with those of the optimum solutions found through a brute force search of the solution space. For the potential sets of task assignments given in Table 6.5, the brute force search of the solution space finds a single optimal task plan for potential task assignments 3 and 5 but finds 6 equally optimal task plans for the other potential sets of task assignments. Table 6.6 shows the highest mean cost of task plans found by DGSA Layer 2 in comparison to those of the optimal task plans found via a brute force search of the solution space for the potential sets of task assignments given in Table 6.5. Table 6.6 shows that DGSA Layer 2 finds optimum or extremely close to optimum task plans for potential task assignments 2 and 6 irrespective of the number of searcher agents, S, and the maximum number of iterations, λ, DGSA Layer 2 uses in its main phase. In contrast, DGSA Layer 2 finds task plans with a slightly higher mean cost than the optimum solutions for potential task assignments 1, 3, 4 and 5 when a small amount of searcher agents, S, are used. The mean percentage difference in costs between the task plans DGSA Layer 2 finds and the optimum task plans for potential task assignments 4, which saw the highest deviation, is shown in Figure 6.10. Figure 6.10 shows that increasing the maximum number of iterations, λ, used in the main phase of the search algorithm has an inconsistent effect on its accuracy. However, increasing the number of searcher agents, S, used in the main phase of its search algorithm

consistently improves its accuracy. Here, S = 10 searcher agents are required to consistently find the optimum or extremely close to optimum task plans regardless of the maximum number of iterations, λ, used in the main phase of the search algorithm. This behaviour is also shown when DGSA Layer 2 is applied to potential task assignments 1, 3 and 5 with the corresponding number searcher agents, S, required to consistently find the optimum or extremely close to optimum task plans given in Table 6.6.

Table 6.7: A table showing a comparison of the highest mean completion time of task plans found by DGSA Layer 2 in comparison to those of the optimal task plans found via a brute force search of the solution space for the potential sets of task assignments given in Table 6.5. Additionally, the number of searcher agents required to reduce mean percentage cost difference to zero regardless of the maximum number of iterations, λ, used in the main phase of the search algorithm are given.

Potential Set of Task Assignments	**Completion Time of Optimum Solutions Found by Brute Force Search (Seconds)**	**Highest Mean Completion Time of Solutions Found with DGSA Layer 2 (Seconds) and Settings Used**	**Highest Mean Percentage Completion Time Increase of Solutions Found with DGSA Layer 2**	**Number of Searcher Agents, *S*, Required to Reduce Mean Percentage Completion Time Difference to Zero**
1	206.8	208.63 $S = 2, \lambda = 10$	0.88%	7
2	180.4	180.4 $S = 2, \lambda = 4$	0%	2
3	191.9	193.1 $S = 2, \lambda = 6$	0.63%	5
4	168.1	169.75 $S = 2, \lambda = 6$	0.98%	10
5	196	196.8 $S = 2, \lambda = 4$ and $\lambda = 10$	0.41%	6
6	163	163 $S = 2, \lambda = 4$	0%	2

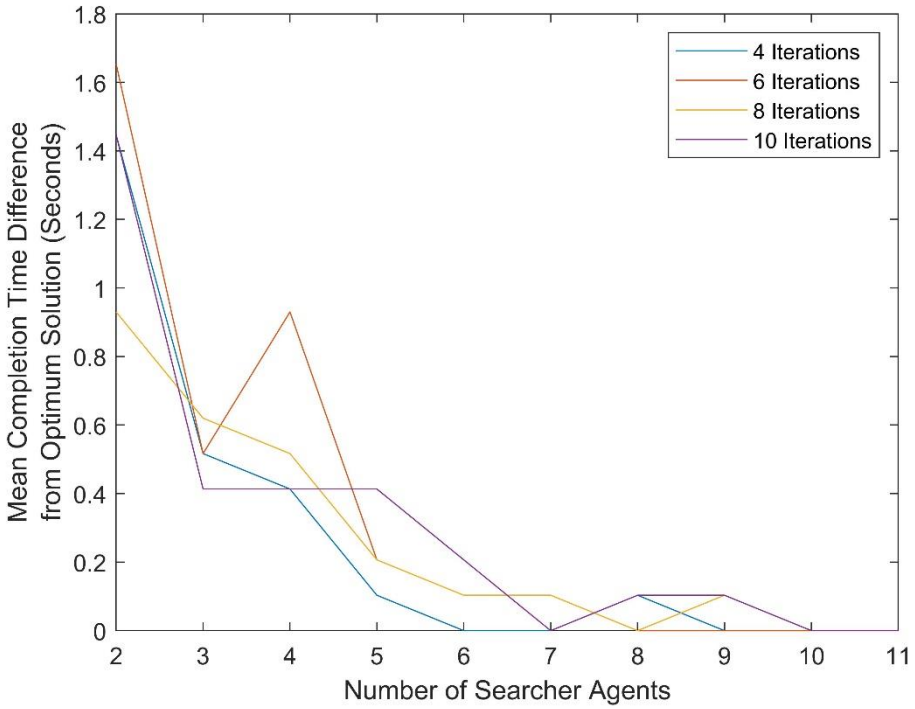

Figure 6.11: Plot of the difference in total completion times between the task plans found for potential task assignments 4 by DGSA Layer 2 and the optimum task plans found using brute force, with varying number of searcher agents and maximum number of iterations used by the main phase of DGSA Layer 2.

To test the accuracy of DGSA Layer 2, it is also necessary to compare the completion time of the solutions DGSA Layer 2 finds for the potential sets of task assignments in Table 6.5, with those of the optimum solutions found through a brute force search of the solution space. Table 6.7 shows the highest mean completion time of task plans found by DGSA Layer 2 in comparison to those of the optimal task plans found via a brute force search of the solution space for the potential sets of task assignments given in Table 6.5. It is again shown that the task plans DGSA Layer 2 finds have a slightly higher mean completion time than the optimum solutions for potential task assignments 1, 3, 4 and 5 when a small amount of searcher agents, S, are used. The highest deviation is again seen for potential task assignments 4, however, the magnitude of this deviation is smaller than that seen for the cost of the task plan. The mean completion time difference between the task plans DGSA Layer 2 finds and the optimum task plans for potential task assignments 4 is shown in Figure 6.11. Figure 6.11 again shows that increasing the maximum number of iterations, λ, used in the main phase of the search algorithm has an inconsistent effect on its accuracy. However, increasing the number of searcher agents, S, used in the main phase of DGSA Layer 2 again consistently improves its accuracy. Table 6.7 shows that the searcher agents required to consistently find the optimum or extremely close to optimum task plans regardless of the maximum number of iterations, λ, used remains unchanged for the potential task assignments in Table 6.5.

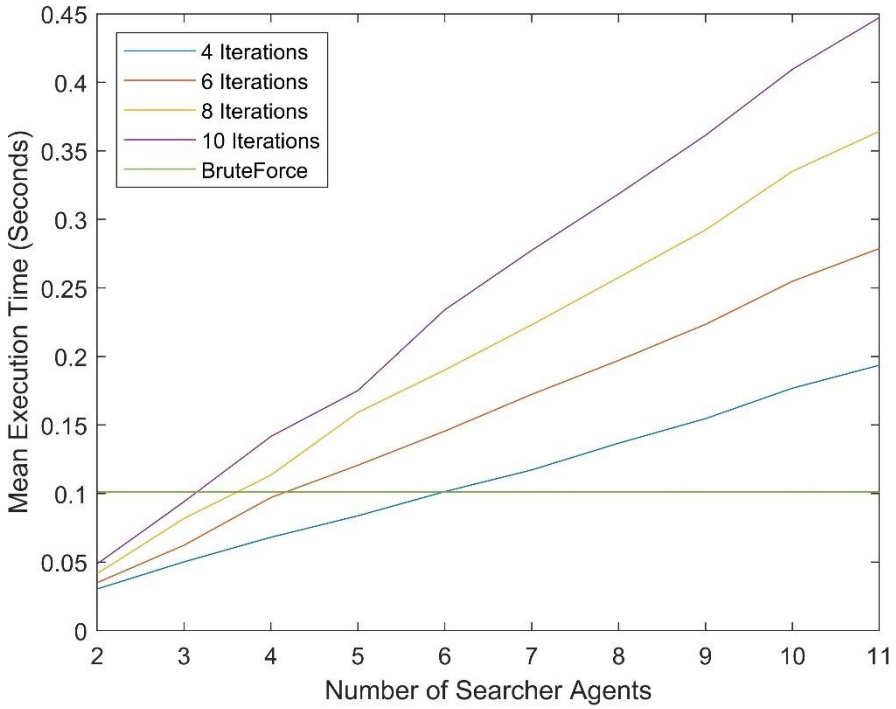

Figure 6.12: Plot of the mean execution time for DGSA Layer 2 when applied to potential task assignments 1, with varying number of searcher agents and maximum number of iterations used by the main phase of DGSA Layer 2 in addition to the execution time of a brute force search of all potential task plans for potential task assignments 1.

To complete analysis of the performance of DGSA Layer 2, it is necessary to assess the corresponding execution time of the search algorithm for each of the potential task assignments in Table 6.5 compared to a brute force search of the solution space. Here, Figure 6.12 shows the mean execution time of DGSA Layer 2 when applied to potential task assignments 1 under the tested settings in addition to the brute force execution time to find the optimum task plan. This shows that, as with DGSA Layer 1, the mean execution time of DGSA Layer 2 displays a linearly increasing trend when increasing the number of searcher agents, S, it uses with the maximum number of iterations in its main phase, λ, significantly affecting the growth rate. The same behaviour is shown when DGSA Layer 2 is applied to each of the potential sets of task assignments in Table 6.5. In all of these cases, increasing the maximum number of iterations, λ, DGSA Layer 2 uses in its main phase has an inconsistent effect on its accuracy compared to increasing the number of searcher agents. This again implies that priority should be given to increasing the number of searcher agents DGSA Layer 2 uses rather than the maximum number of iterations it uses in its main phase.

For potential task assignments 1, 3, 4 and 5 a mean number of $S = 7$ searcher agents is required to find an optimum or extremely close to optimum task plan in all 30 runs of DGSA Layer 2 regardless of the maximum number of iterations used in its main phase. This mean value does not consider the

accuracy for potential task assignments 2 and 6 since in those cases an optimal task plan was found in all 30 runs regardless of the settings used in DGSA Layer 2. Table 6.8 shows the mean execution times of DGSA Layer 2 when using these optimum settings of S = 7 searcher agents and a maximum number of λ = 4 iterations in the main phase of DGSA Layer 2, along with the execution times of a brute force search of all potential task plans. Under these settings DGSA Layer 2 is slower by a mean of 23%, however, it is important to note that this increase only represents a few hundredths of a second. Although DGSA Layer 2 is slower in this case, it is important to note that for larger solution spaces the execution time of the brute force method would not scale as efficiently as that of DGSA Layer 2. As seen with the testing of the task planner utilising DGSA Layer 1 only, in larger solution spaces the Discrete Gravitational Search Algorithm is a far more efficient way of finding a good task plan for a set of task assignments.

Table 6.8: A table of the execution times of a brute force search of all potential task plans for a proposed set of task assignments (given in Table 6.5) in addition to the mean execution time of 30 runs of DGSA Layer 2 under the proposed optimum settings. A comparison of the execution times of both methods is also presented with the mean increase in execution times and the corresponding percentage increase.

Potential Set of Task Assignments	**Execution Time of Brute Force Method (Seconds)**	**Mean Execution Time of DGSA Layer 2 (Seconds) when S = 7 and λ = 4**	**Mean Increase in Execution Time (Seconds)**	**Mean Percentage Increase in Execution Time**
1	0.1012	0.1172	0.016	15.8%
2	0.0987	0.1215	0.0228	23.1%
3	0.0970	0.13	0.033	34%
4	0.0994	0.1161	0.0167	16.8%
5	0.0954	0.1276	0.0322	33.8%
6	0.1085	0.1244	0.0159	14.7%

6.4.4. Testing of the Full Dynamic Task Planner

Following the tests to determine the accuracy and runtime of DGSA Layer 2 it is necessary to apply the full dynamic task planner to the complex manufacturing task to determine its accuracy and execution time. For DGSA Layer 1 and DGSA Layer 2 the same settings are applied as those used in the testing contained within Sections 6.4.2 and 6.4.3, respectively. In this case, the optimum number of searcher agents, S, and maximum number of iterations, λ the main phase of the search algorithm uses are set based on the results of the experiments in Sections 6.4.2 and 6.4.3. Here DGSA Layer 1 utilises S = 70 searcher agents over a maximum number of λ = 25 iterations of its main phase whereas DGSA Layer 2 utilises S = 7 searcher agents over a maximum number

of $\lambda = 4$ iterations. Again, the dynamic task planner is run 30 times to determine accuracy of the solution and execution time compared to the optimum set of task assignments and task plan generated using a brute force search of all possible task plans for every possible set of task assignments. This brute force search was again achieved by generating all possible sets of task assignments and task plans that satisfy the result of the task pre-execution constraints then applying the objective function to each of them to generate a data set. The set of task assignments and task plans with the minimum cost was found by searching this data set using built in Minimum functions of the MATLAB software package. Here, the dynamic task planner and the brute force search of the solution space are once again both implemented in the MATLAB software package. The mean cost and total completion time of the solutions generated by the dynamic task planner are shown in Table 6.9 alongside those of the optimum solution obtained through a brute force search of the solution space. Table 6.9 also shows the mean execution time of the dynamic task planner in addition to the execution time of the brute force search of the solution space.

It is first important to note that the brute force search of the entire solution space finds a single optimum set of task assignments with 6 optimal task plans. Analysing Table 6.9 shows that the costs of the sets of task assignments and task plans found by the dynamic task planner are a mean of 17.1% higher than those of the optimum solution. Additionally, the mean difference in completion times from the optimal solution is 6.27 seconds meaning they are approximately 4.7% larger. For the 30 runs of the complete dynamic task planner in this task planning problem the mean execution time is approximately 22.4 minutes which is approximately 1.05% of the execution time of the brute force search.

Table 6.9: A table of the mean cost and completion time of the solutions generated by the dynamic task planner for the complex task as well as those of the solution generated through a brute force search of the solution space. In addition to this the mean execution time of the dynamic task planner and the execution time of the brute force method are presented.

Method	**Cost of Solution**	**Completion Time of Solution (Seconds)**	**Execution Time to Calculate**
Brute Force Search	5.61	133.5	34.9 Hours
Dynamic task planner (Mean Value of 30 Runs)	6.57	139.77	22.4 Minutes

Here, a better accuracy is achieved for the full dynamic task planner compared with when only DGSA Layer 1 is utilised with a few pre-set possible task plans. Despite this it is shown that the optimal set of task assignments and

task plans are still not found by the dynamic task planner, but the solutions found are still good solutions. In this case, the dynamic task planner has an execution time of 22.4 minutes meaning a new set of task assignments and task plan could not be generated after a task iteration and be implemented in the next iteration. Instead in cases such as this where both layers of the task planner are implemented, the dynamic task planner would be required to operate whilst the task continues to be executed by the HR team with the resulting task assignments and task plan being implemented when ready.

6.5. Chapter Summary

In this chapter a methodology was presented for generating new sets of task assignments and task plans for HR teams in a manufacturing task. This methodology utilised the Discrete Gravitational Search Algorithm (DGSA) proposed in (Dowlatshahi, Nezamabadi-Pour and Mashinchi, 2014), a metaheuristic search algorithm that allowed the solution spaces of the task planning problem to be searched quickly to find a good solution. The Discrete Gravitational Search Algorithm was implemented in a dual-layer dynamic task planner with the first layer, DGSA Layer 1, searching for possible task assignments for a HR team and the second layer, DGSA Layer 2, searching all possible task plans for each potential set of task assignments as shown in Figure 6.3.

To test the accuracy and speed of this task planning methodology in addition to determining the optimal settings for each layer of the DGSA, it was applied to an example assembly task for a turbocharger given in (Nikolakis et al., 2018) that was capable of being performed by a HR team. Given the task precedence relations for this assembly task, there were only two possible task plans, meaning that the second layer of the task planner, DGSA Layer 2, could not be implemented. To allow the full dynamic task planner to be tested, two example assembly tasks were simulated based on the turbocharger task given in (Nikolakis et al., 2018). The first task, the simple task, consisted of the unaltered turbocharger task whilst the second task, the complex task, consisted of the same subtasks with a simulated set of precedence relations for the completion of the subtasks allowing for multiple potential task plans.

For the simple assembly task, the dynamic task planner was applied with DGSA Layer 2 replaced by a brute force check of the two possible task plans. In this case the accuracy of the task planner meant that the found sets of task assignments and task plans had a mean cost around 39% larger than the solution found using a brute force search of the solution space with a mean task completion time approximately 30 seconds larger under optimal settings. Despite this accuracy, the task planner could be executed in a mean execution time of 1 to 2 seconds, compared to 11.5 minutes for the brute force search, meaning it could be executed after a task iteration with the new set of task

assignments and task plan being implemented in the next task iteration. Such performance enables the viability of the overall task planning methodology proposed in Chapter 3, allowing the task planner to be implemented at set intervals to ensure the current set of task assignments and task plans for a HR team reflects the current capabilities of each worker.

For the complex task, it was first necessary to determine the accuracy and speed of DGSA Layer 2 to find an optimum task plan for six potential sets of task assignments in addition to determining its optimal settings. In these cases, it was shown that DGSA Layer 2 was consistently capable of finding an optimal or very close to optimal task plan given a set of task assignments, regardless of the settings used. Through testing with these six potential sets of task assignments the optimal settings for DGSA Layer 2 were determined that balanced accuracy of the solution with minimising the execution time of the search algorithm. Under these optimal settings, DGSA Layer 2 had a mean execution time of 0.12 seconds which was on average 23% slower than a brute force search of the solution space. Although slower, it was important to note this difference in execution time was a few hundredths of a second and that DGSA Layer 2's execution times would scale far more favourably with larger solution spaces than brute force methods.

Once the optimal settings for DGSA Layer 1 and DGSA Layer 2 were determined, the full dynamic task planner was applied to the complex task to determine its accuracy and speed under these settings. It was shown that the dynamic task planner had a better accuracy than in the simple task with the found sets of task assignments and task plans having a mean cost that was 17.1% higher and a mean total completion time that was 6.3 seconds slower than those of the optimal solution. Given the much larger solution space of this problem, the mean execution time of the dynamic task planner was 22.4 minutes compared to the execution time of 34.9 hours for a brute force search of the solution space. The large magnitude of the execution time of the dynamic task planner in the complex task compared to that in the simple task would mean that the task could not be re-planned between task iterations of the manufacturing task. To enable the viability of the overall task planning methodology proposed in Chapter 3 for such complex tasks, it would be required for the task to be planned in the background whilst the human and robot workers continue to execute the task and implement the new set of task assignments and task plan when it is ready. Given the optimal completion time of the complex task of 133.5 seconds, it could take 10 to 11 iterations of the task before the new task plan would be ready and potentially mean the new task plan could lose its relevance. It is important to note that the execution time could be reduced with optimisation of the code executing the dynamic task planner and better hardware. However, given the magnitude of the completion time this would likely still require the dynamic task planner to be executed in this way.

7. Utilising Dynamic Task Planning to Replan Manufacturing Tasks Across a Work Shift

7.1. Introduction

Following the introduction of the task planning methodology and tests of its accuracy in finding an optimum set of task assignments and task plan, it is necessary to test the effects of dynamic task planning across a work shift to determine the benefits that it can bring to optimising the use of Human-Robot (HR) teams. In this research, work shifts are simulated to allow the testing of multiple scenarios of human performance reflecting variation in their capabilities and enabling analysis of how the task planner would react in these situations. Two groups of scenarios are considered, the first of which relates to changes in performance of a human worker detectable by continuous variables described in Chapter 4. The second of these groups of scenarios relates to changes in capabilities of the human and robot workers detectable by discrete variables described in Chapter 5. These scenarios are tested with both the simple and complex tasks described in Chapter 6 to determine the utility of task planning, respectively, for the existing linear production task, described in Section 6.4.1, in addition to the simulated more modular production task, described in Section 6.4.3.

This chapter begins with the definition of methods for determining when task assignments should be re-evaluated in Section 7.2. To achieve this, methods are first defined to determine the intervals between task replanning attempts for the re-evaluation of task assignments and plans. Following this, a checking function is defined to determine whether the change in worker performance or capabilities is large enough to justify task replanning. Next, the pre-execution constraints for the dynamic task planner are modified to ensure that the pre-allocation of tasks does not restrict the capabilities of the dynamic task planner. Additionally, modifications are defined to handle situations that could potentially be problematic for the dynamic task planner. This chapter continues with the detailed the setup of simulated work shifts for a HR team in Section 7.3 to allow testing of the dual-layer dynamic task planner. To achieve this, methodologies are defined for generating simulated production data for human and robot workers which are then used to generate costs for the workers to complete subtasks of the overall manufacturing task. To complete the setup of the simulations, it is necessary to define the structure of the simulated work shift to dictate the length of the work shift and when the human worker should take breaks. Finally, the results of these simulations are discussed in Section 7.4, analysing the effect of task replanning on the efficiency of the simulated HR team.

7.2. Task Assignments and Plan Re-Evaluation

7.2.1. Determining Intervals Between Replanning Attempts

The task planning system proposed in this research is designed to re-evaluate sets of task assignments and task plans at set intervals during a work shift to minimise the increase in cost for the task plan over a work shift whilst optimising the efficiency of the HR team. This is necessary as continuously replanning task assignments over the course of a work shift with constant switching of tasks would likely cause confusion in a real-life HR team and negate the benefits of task replanning. The set intervals between task replanning must be defined by the end user, by specifying the number of iterations of the full manufacturing task completed before re-planning can occur. This definition allows re-planning attempts to be made at regular intervals whilst making the intervals easily definable by the end user.

Due to the limitations of the dynamic task planner, it is necessary to define two separate procedures for instigating task replanning opportunities. This is necessary due to the increasing magnitude of execution time for the dynamic task planner with the increasing complexity of the task assignment and planning problem. It was shown in Chapter 6 that for the simple task, where DGSA Layer 2 was not employed, that the dynamic task planner had an execution time of a few seconds. However, for the complex task, when using both DGSA Layers 1 and 2, the full task planer could take up to 22 minutes. This means that for the simple task, the task planner can replan the task between task iterations, whereas for the complex task, the task must be replanned whilst production continues with the new set of task assignments and task plan being implemented when ready.

The procedure for scheduling replanning attempts for the simple task operates by instigating attempts at set intervals of 10 task iterations with this pattern broken only by worker break periods. For these break periods, replanning attempts should occur after the last task iteration before the human worker returns to the manufacturing task to ensure a new set of task assignments and task plan is ready for their return. The following replanning attempts should occur every 10 iterations from this new last occurrence of replanning. A diagrammatic representation of this scheduling procedure is shown in Figure 7.1.

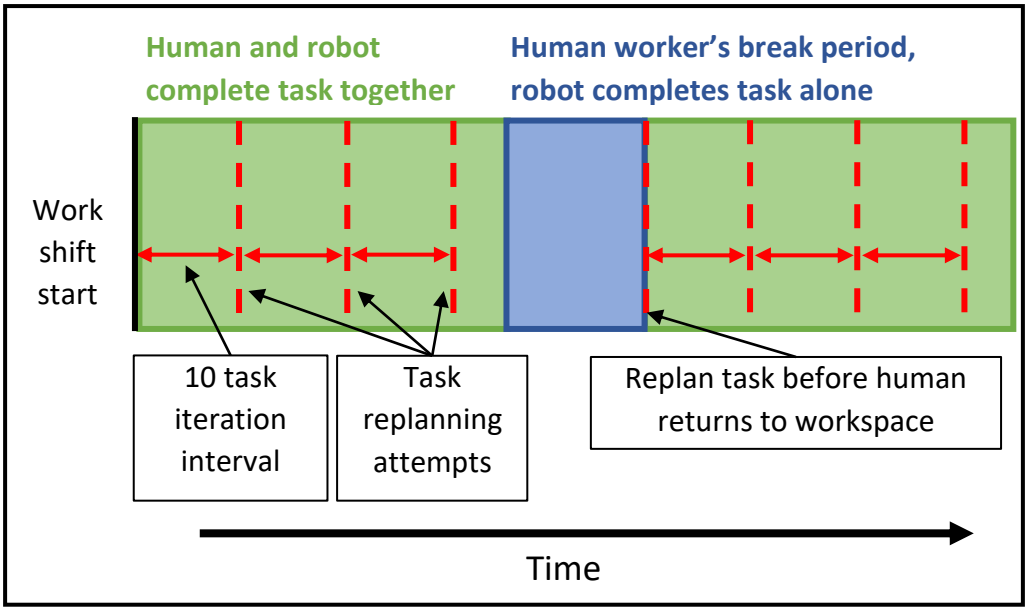

Figure 7.1: A representation of the procedure for scheduling task replanning attempts for the simple manufacturing task. This illustrates the methodology of scheduling task replanning at intervals of 10 complete task iterations with the set of task assignments and task plan being implemented in the next task iteration. An exception to this procedure occurs with a break period for a human worker, where the task must instead be replanned after the last task iteration before the human worker returns to the manufacturing task. Task replanning then continues at the set interval of 10 task iterations from this point.

The procedure for scheduling replanning attempts for the complex task instead operates by instigating replanning attempts at set intervals of 10 task iterations from the implementation of the last set of task assignments and task plan. This is necessary to ensure that there is a consistent amount of data from each implemented set of task assignments and task plan available to generate costs for use in the next attempt at replanning. In addition to this, to ensure a new set of task assignments and task plan is ready when the human worker returns from their break period, it is necessary to ensure replanning is instigated with enough time for the task planner to be executed. To achieve this, it is necessary to instigate a replanning attempt when the time difference between the human worker's remaining break period and the mean execution time of the task planner in the current work shift is less than twice the current execution time of the manufacturing task. A diagrammatic representation of this scheduling procedure is shown in Figure 7.2.

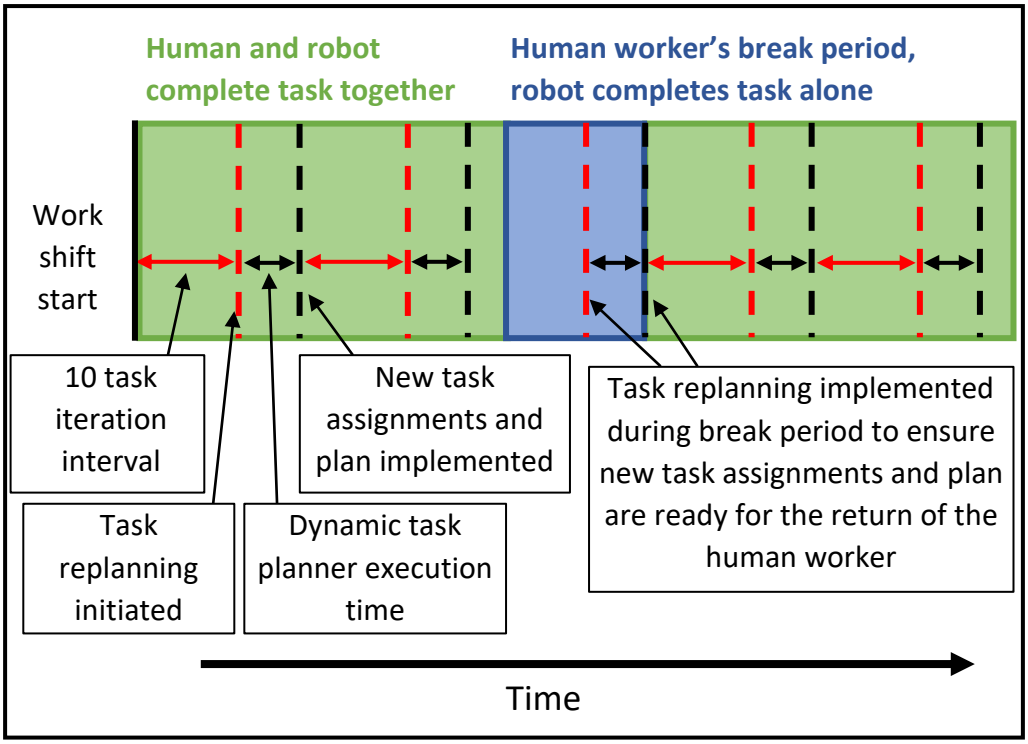

Figure 7.2: A representation of the procedure for scheduling task replanning attempts for the complex manufacturing task. This illustrates how the methodology of scheduling task replanning at intervals of 10 complete task iterations must be modified to account for the larger execution time of the dynamic task planner when applied to the complex manufacturing task. Here task execution by the HR team must continue whilst the task is replanned with a new set of task assignments and task plan implemented when they are ready. To accommodate this, the next task replanning attempts should occur at an interval of 10 complete task iterations after the last set of task assignments and task plans was implemented. Additionally, during the human worker's break period, it is necessary to ensure replanning is instigated with enough time for the dynamic task planner to be executed so a new set of task assignments and task plan is ready for their return.

7.2.2. Replanning Utility Checking Function

Despite instigating task replanning attempts at set intervals, it may not be necessary to execute replanning if worker costs have not significantly changed. In this way, unnecessary computational expense to find the same or similar set of task assignments and task plan is avoided. Additionally, in the cases of multiple optimum task assignments and plans, replanning could result in task assignment and task plan changes that do not reduce the cost for the HR team to complete the manufacturing task or improve its efficiency. To this end, it is desirable to introduce a checking function based on worker costs that determines whether it is necessary to execute task replanning when a replanning attempt is instigated. This checking function must determine if there has been any significant change in worker costs, by assessing the current costs for each worker against their costs from the last occurrence of task replanning. The function is used to check several conditions which, if any are satisfied, determine if the set of task assignments and task plan should be re-evaluated. These conditions are designed to implement the task planner in cases where replanning could potentially be appropriate to improve the

efficiency of the HR team and if this is not possible minimise cost increase across the work shift.

The first of these conditions handles cases where the optimal worker for a subtask has changed between the current replanning attempt and the last occurrence of task replanning. This situation necessitates replanning as a change in the optimal worker for a subtask could result in a different set of task assignments with a smaller overall cost to carry out the manufacturing task. To assess such a change, it is necessary to define the optimal worker for each subtask as the worker with the lowest cost to complete the subtask. This list of optimum workers must then be reassessed with each task replanning attempt. If the optimum worker has changed for one or more subtasks since the last occurrence of task replanning then the condition for task replanning to be executed is fulfilled.

The second of these conditions handles cases where the cost for a worker to complete a subtask has increased significantly from their cost at the last occurrence of task replanning. This situation necessitates task replanning, even if the task assignment remains optimal, as a new task plan with a different ordering of subtasks could reduce the overall cost for the HR team to complete the manufacturing task. To assess this, the difference in cost between the current iteration of the task and the costs from the last occurrence of task replanning is calculated for each worker in each subtask. To determine if the task should be replanned a threshold cost difference, ω, must be defined to determine how significant the change in worker costs should be to warrant a task replanning attempt. If any of the cost differences are greater than a threshold cost difference, ω, then the DGSA task planner should be executed.

Combining these procedures for these re-planning conditions gives the full checking function and the fulfilment of either of these conditions results in the checking function returning a value of 1. If this occurs, the task can then be re-planned with the dynamic task planner proposed in Chapter 6. However, if the checking function returns a value of 0 then task replanning is not carried out to avoid unnecessary computational expense.

7.2.3. Modification to Task Planner Pre-Execution Constraints

To enable optimal application of the dynamic task planner across a work shift, it is necessary to make several modifications to the task planner pre-execution constraints proposed in Section 6.3. This is necessary due to the operational limits of using online data to generate costs for human and robot workers to complete manufacturing subtasks, in addition to ensuring the dynamic task planner is not restricted by excessive pre-allocation of subtasks.

First, it is necessary to mitigate for a possible task planning scenario where a human worker is assigned a subtask but has not yet completed this

subtask in the current work shift. Such a situation would result in there being no data from the current work shift relating to their current performance and capabilities in completing this subtask, meaning an accurate cost could not be calculated for the subtask to allow a fair comparison with the robot worker. Second, it is necessary to ensure that the task planner pre-execution constraints do not restrict the capabilities of the dynamic task planner to search other task assignments through excessive pre-allocation of subtasks before use of the dynamic task planner. This is necessary when human workers become over or under fatigued as it is more likely that the cost difference between them and the next best worker will increase beyond the threshold cost difference, η, defined for this procedure unless it is set very high.

These operational limits are mitigated by modifying the pre-execution constraints in Section 6.3 to ensure that the task plan pre-execution constraints cannot pre-assign more than 50% of the manufacturing subtasks to a worker before the dynamic task planner is used. This is applied by instead generating the pre-execution constraints with the following procedure:

1. The locked in identifier, ψ_i, is set as 1 for a subtask *i* in the list of locked in task assignments, Ψ, if it was assigned to the robot worker in the initial set of task assignments. The corresponding task assignments, α_i, in the corresponding set of base task assignments, Λ, are set as the worker number for the robot worker.

2. The remaining percentage of pre-allocated subtasks are assigned as in the task pre-allocation procedure defined in Section 6.3 with the exception that assignment priority will be defined by the magnitude of the difference in cost between a worker and the next best worker in descending order from the maximum cost difference.

The first step in this procedure ensures that tasks cannot be reassigned to human workers during the work shift if they were assigned to a robot worker in the initial set of task assignments generated at the start of the current work shift. This step could only be overridden if the robot worker's cost for a manufacturing subtask increases significantly between task iterations for a task locked in for them. This would allow the task to be reassigned to a human worker as it indicates the robot worker's capabilities have significantly decreased. However, the consistency of robot completion times in completing tasks means this scenario would only occur if a robot worker made significant errors in completing a subtask resulting in the cost increasing for discrete variables in that subtask. Due to the accuracy and tolerance to repetitive tasks of robot workers, such a scenario is assumed to have a low probability of occurrence and is not considered within this research. The second step of this procedure allows tasks to be locked in as with the pre-execution constraints in

Section 6.3. However, it also ensures that pre-allocation of subtasks does not restrict the search space of the dynamic task planner.

7.3. Dynamic Task Planner Testing: Replanning Sets of Task Assignments and Task Plans across a Work Shift

7.3.1. Simulating Work Shifts for Human and Robot Workers

To test the effect of task replanning on the efficiency of the HR team, it is decided to simulate the performance of a HR team completing a manufacturing task across a work shift with and without implementation of the dynamic task planner during a work shift. In both of these cases, the initial set of task assignments and task plans for the HR team are generated by utilising the historic data on worker performance as demonstrated in Chapter 6. This allows testing of the dynamic task planner in multiple potential scenarios of changes to the performance and capabilities of human and robot workers. This testing is again conducted by applying the dynamic task planner to the simple and complex example manufacturing tasks from Chapter 6 utilising one human and one robot worker, again defined as workers 1 and 2, respectively. Costs for these workers are generated using the full cost functions proposed in Section 6.4.1 including the fatigue and precision of sealant application variables. To simulate these work shifts, it is necessary to generate the input data for the cost functions in addition to setting the variables for their calculation and defining the structure of the work shift in terms of work and break periods. This setup is defined over Section 7.3 beginning with the simulation of completion times for subtasks a worker is currently assigned followed by estimating what their completion times would be when they are not assigned a subtask. Next the generation of costs for workers across the simulated work shift is defined followed by the setup of the simulated work shifts and which scenarios of worker performance and capabilities these are designed to simulate.

7.3.2. Simulating Completion Times for Tasks Assigned to a Worker

To generate costs for the human worker using the full cost functions from Section 6.4.1, it is first necessary to generate input data for the fatigue and completion time variables defined in Chapter 4. To apply the fatigue variable, it is first necessary to define how expected completion times for the human are generated for each of the subtasks assigned to the human worker in the example manufacturing tasks. As defined with the fatigue variable in Chapter 4, the expected completion time, $E_{i,j}$, for the human worker in task iteration i of subtask j is generated using the completion time model proposed by (Digiesi et al., 2009) as given by Eqs. (4.1), (4.2) and (4.4) for subtasks the human worker is assigned. To apply this model, it is first necessary to determine

the historic initial completion time for the human worker in a task assignment period and generate the synthetic measure of fatigue, τ_j', for a subtask *j* via Eq. (4.4). In both example manufacturing tasks, the initial completion times, $t_{1,j}$, of subtask *j* for the human worker are given by the historic initial completion times in Table 6.1. This initial completion time must change if the task is un-assigned and later reassigned to the human worker to the initial completion time of the current task assignment period, as defined by Eq. (4.2).

To generate the synthetic measure of fatigue it is necessary to know how many iterations, D_j, of a subtask *j* the human worker can complete in the time period of length T_j seconds. It is assumed that the manufacturer implementing this system will have or is able to obtain this information for each subtask of a manufacturing task for a human worker. In the case of these simulated work shifts, it is necessary to generate this information as this was not provided in (Digiesi et al., 2009). In this research it is assumed that the human worker, when performing as expected, can complete 75% of the task iterations that would be completed if they completed the task consistently with their historic initial completion time as given in Table 6.1. This allows the definition of the number of iterations of a subtask *j* that the human worker can complete in time period T_j as

$$D_j = \left\lfloor 0.75 \frac{T_j}{t_{w,j}} \right\rfloor. \tag{7.1}$$

In this research, D_j is defined for each subtask by setting the time period T_j as one hour as it is assumed that a manufacturer would give this information for a pre-specified time period. Using the initial completion time $t_{w,j}$ for a task assignment period of subtask *j* beginning in task iteration *w*, along with the number of iterations, D_j, of the subtask completed in a time period of T_j = 3600 seconds it is possible to calculate the synthetic measure of fatigue τ_j' using Eq. (4.4).

To apply the fatigue variable for human workers and completion time variable for both workers defined in Chapter 4, it is also necessary to simulate the completion times depicting those collected as workers are performing their assigned subtasks. In this research, the initial completion times for the robot worker given in Table 6.1 are used to simulate their completion times across a work shift. These completion times are used as it is assumed that a robot worker's completion times would vary minimally compared to those of a human worker due to their accuracy and repeatability. In contrast, for the human worker it is decided to simulate three possible scenarios of human performance to test the capabilities of the dynamic task planner. These scenarios include the human worker performing as expected, in addition to where the human worker is over or under fatigued representing worse or better than expected performance, respectively. When over or under fatigued,

the human worker's costs will increase or decrease significantly due to the fatigue variable in addition to their completion times affecting the potential idle time of workers. These changes in performance mean that the optimal set of task assignments and task plan could change significantly compared to when the human worker is performing as expected.

To generate completion times for these scenarios, the completion time model proposed by (Digiesi et al., 2009) is again used with over or under fatigued performance being simulated by attenuating the historic initial completion time of the human worker for each subtask. In the example case in this research, it is assumed that when the human worker is over fatigued or under fatigued that their initial completion time for a subtask at the beginning of a work shift will increase or decrease, respectively, by 10%. Using this assumption, the initial completion times for each subtask in the initial iteration of the simulated work shift for each of these scenarios are given in Table 7.1 in addition to the resulting number of iterations, D_j, of each subtask completed in a time period of T_j = 3600 seconds generated with Eq. (7.1). These values allow the synthetic measure of fatigue, τ_j', to be generated for each subtask in each of the fatigue scenarios using Eq. (4.4) and subsequently generate the completion times across a work shift using Eq. (4.1).

It is also necessary to add noise to the simulated real completion times to simulate the natural variation in human completion times that would be present in real data instead of the smooth increase in completion times generated from the model proposed by (Digiesi et al., 2009). In this research, it is assumed that this natural variation in the completion time for a subtask for the human worker is at most ±5% of the initial completion time for the subtask in the first iteration of the simulated work shift. Since this is considered at the extremes, the actual variation for the human worker's completion times in a subtask *j* is generated as $0.05\ t_{1,j}$ multiplied by a random variable generated from the N(0,0.125) distribution. This distribution is chosen to ensure that the extremes of the simulated variation in completion time can be reached but not exceeded. This simulated natural variance in completion times is only added from the second iteration onwards of every task assignment period.

Table 7.1: A table of the initial completion times and number of iterations of the subtask completed in a time period of Tj = 3600 seconds for the human worker under the three proposed fatigue scenarios.

Subtask	Human Performing as Expected		Under Fatigued Human Worker		Over Fatigued Human Worker	
	Initial Completion Time, $t_{1,j}$	Number of Iterations Completed, D_j	Initial Completion Time, $t_{1,j}$	Number of Iterations Completed, D_j	Initial Completion Time, $t_{1,j}$	Number of Iterations Completed, D_j
1	6	450	5.4	500	6.6	409
2	4.7	574	4.23	638	5.17	522
3	16.6	162	14.94	180	18.26	147
4	8.1	333	7.29	370	8.91	303
5	3.9	692	3.51	769	4.29	629
6	1.7	1588	1.53	1764	1.87	1443
7	7.9	341	7.11	379	8.69	310
8	18	150	16.2	166	19.8	136
9	19	142	17.1	157	20.9	129
10	5.3	509	4.77	566	5.83	463
11	2.4	1125	2.16	1250	2.64	1022
12	1.7	1588	1.53	1764	1.87	1443
13	2.7	1000	2.43	1111	2.97	909
14	3.3	818	2.97	909	3.63	743
15	8.5	317	7.65	352	9.35	288
16	8.8	306	7.92	340	9.68	278
17	13.5	200	12.15	222	14.85	181
18	9.5	284	8.55	315	10.45	258
19	1.9	1421	1.71	1578	2.09	1291
20	2.4	1125	2.16	1250	2.64	1022
21	7.2	375	6.48	416	7.92	340
22	12.3	219	11.07	243	13.53	199
23	17.7	152	15.93	169	19.47	138
24	2.6	1038	2.34	1153	2.86	944

7.3.3. Simulating Completion Times for Tasks Not Assigned to a Worker

To generate costs for the workers using the full cost functions from Section 6.4.1, it is also necessary to generate input data for the fatigue and completion time variables for subtasks that are not currently assigned to a worker. Once a subtask is taken away from a worker it is necessary to model how their performance should change to determine their cost for completing the task. By determining costs in this way for subtasks the worker is no longer assigned, this enables the task planner to determine if a worker's performance should have improved since the subtask was reassigned and possibly allow the subtask to be reassigned back to them.

Here, the initial completion times for the robot worker given in Table 6.1 are also used to simulate their completion times across a work shift when they are not currently assigned a subtask. This is possible as the assumption, in Section 7.3.2, that a robot worker's completion times would vary minimally compared to those of a human worker still holds in such cases. To generate expected and current simulated completion times for a human worker, it is instead necessary to define a recovery completion time model that determines how a worker's completion times for a task would improve when they are not assigned a subtask of the overall manufacturing task. Through a literature review, a counterpart model to the fatigue model proposed in (Digiesi et al., 2009) for recovery of completion times in a repetitive manufacturing task was not found. Due to this and to provide a recovery model required for simulating a work shift, it is assumed that a human worker's completion times would recover at the same rate that they became fatigued, allowing the growth factor of the model proposed in (Digiesi et al., 2009) to be reversed. As this recovery model follows the same approach as the model proposed in (Digiesi et al., 2009) it must be calculated individually for each subtask that is not currently assigned to a human worker with the exception of subtasks the human worker has never been assigned in the current work shift.

To generate the recovery model, it is first necessary to define an initial completion time from which the completion times will reduce as the human worker recovers. Since this recovery is from the completion time at the end of the last task assignment period, the initial completion time for a subtask is set as the expected completion time, $E_{\Theta_j,j}$, where Θ_j is the last task iteration that the human worker was assigned subtask *j*. Following this, it is necessary to define the synthetic measure of fatigue for use by the recovery model as the factor determining the magnitude of recovery in completion times. The synthetic measure of fatigue, τ'_j, for subtask *j*, used in the last task assignment period of the subtask cannot be used in this case and instead a new synthetic measure of fatigue recovery must be defined. This is necessary as using the synthetic measure of fatigue for a human worker defined in Eq. (4.4) would only allow the completion time for a human worker in a subtask to recover to the completion time of the first iteration of the last task assignment period of the subtask for the human worker. In this research, it is instead assumed that given enough time a human worker's completion times can recover to their initial completion times from the start of the work shift.

To determine this new synthetic measure of fatigue recovery, τ^-_j, for a subtask *j*, it is assumed that recovery will reverse the cumulative fatigue gained over the whole work shift. This requires the recovery model to ignore previous recovery periods and determine the overall rate of fatigue for a worker over the entire work shift. This allows the new synthetic measure of fatigue recovery to be determined using the initial expected completion time for a subtask from

the current work shift and the expected completion time from the last iteration of the last subtask assignment period. Using the justification provided in Appendix A, it is possible to define the synthetic measure of fatigue recovery for the recovery model as

$$\tau^{-}_{j} = \frac{(E_{\Theta_j,j} - E_{1,j})}{\ln(1 + \Theta_j)} \tag{7.2}$$

for a subtask *j*, where $E_{1,j}$ is the worker's expected completion time during the first task iteration and $E_{\Theta_j,j}$ is their expected completion time in the task iteration, Θ_j, that they were last assigned the subtask.

Finally, to generate the recovery model it is necessary to calculate the number of iterations, Γ_j, of the entire manufacturing task that would have been completed over the work shift in the length of time that a human worker has not been assigned a subtask *j* at the current task iteration. To calculate this, the total time elapsed during the work shift is stored after every task iteration to enable calculation of the time elapsed since a worker was last assigned a subtask and determine the task iteration at which the total work shift completion time exceeds this. This enables the expected completion time for a subtask *j* a human worker is not currently assigned to be defined as

$$E_{i,j} = E_{\Theta_j,j} - \tau^{-}_{j} \cdot \ln(\Gamma_j). \tag{7.3}$$

7.3.4. Generating Discrete Capability Data for Human and Robot Workers

In addition to production data for continuous variables, it is necessary to generate production data from discrete variables to assess the task planner's ability to react to discrete instantaneous changes in a worker's capabilities. A barrier to this occurs as the details of quality assurance methods or error checking for subtasks that would be used to form discrete variables were not provided in (Nikolakis et al., 2018) for the turbocharger assembly task. As a result, it is necessary to simulate a discrete variable to determine how the dynamic task planner would react to discrete events causing a sudden increase in cost for a human worker. To achieve this, as described in Section 6.4.1, the discrete precision of sealant application variable from Chapter 5 is used, along with costs generated for the individual sealant errors of the human worker, to represent a typical discrete variable in the cost functions for workers in the simple and complex manufacturing tasks. This variable is applied to the pick and place subtask in addition to the sensing subtask to provide a representative cost for a discrete variable.

To apply the discrete precision of sealant application variable, it is first necessary to simulate the cost of individual iterations of the sealant application subtask when completed within tolerances. These costs are again generated from the truncated normal distributions given in Section 5.4.1 in these cases. This means costs for the human worker for these individual iterations of the sealant application subtask are again randomly generated from a $N(0.05,0.01^2)$ distribution truncated between 0 and 0.1. The costs for the robot worker for these individual iterations are instead randomly generated from an $N(0.02,0.005^2)$ distribution also truncated between 0 and 0.1.

As with the continuous cost function variables, two scenarios are generated where the capabilities of the human worker deviate from their nominal levels. These scenarios follow the format of the third group of scenarios from Section 5.4.1 where a human worker's capabilities in completing the subtask degrade rapidly would resulting in errors that built in severity with each occurrence. The series of discrete errors that formed these scenarios are again those presented in Chapter 5 with Table 7.2 detailing the errors that occur and the iterations they occur in for the human worker.

Table 7.2: A table of the simulated error data for a human worker used for the simple and complex manufacturing tasks to test the reaction of the dynamic task planner to a rapid degradation in a human worker's capabilities in completing the subtask. These scenarios follow the format of the third group of scenarios from Section 5.4.1.

Error Scenario	**Error Generation Data**				
3 Occurrence Set 1	**Task Iteration Error Occurred in**	6	7	8	9
	Error that Occurred	26% Area	25% Length	20° Angle	20° Angle
3 Occurrence Set 2	**Task Iteration Error Occurred in**	12	13	14	15
	Error that Occurred	26% Area	25% Length	20° Angle	20° Angle

7.3.5. Generating Costs for Human and Robot Workers

To generate the dynamic cost functions for each of the workers and their cost for each of the subtasks of the overall manufacturing task, it is necessary to define the required parameters for each of the cost function variables in addition to their weightings. First, the completion variable is defined using the same parameters as used to generate the initial set of task assignments and task plan in Section 6.4.1 to maintain consistency. This is also the case for the discrete precision of sealant application variable which uses

the same parameters given in Section 5.3. For the fatigue variable it is necessary to define new parameters for these simulated work shifts. Here the expected completion time, $E_{i,j}$, for the human worker in iteration *i* of subtask *j* used by the fatigue variable are defined as described in Section 7.3.2 and Section 7.3.3. Additionally, the tolerance to the natural variation in human completion times, $h_{i,j}$, for the human worker in iteration *i* of subtask *j* is defined as 5% of the expected completion time, $E_{i,j}$, for the human worker for simplicity. Finally, for the fatigue variable, the maximum acceptable percentage, e_j, increase or decrease in completion times from the expected completion time, $E_{i,j}$, is again defined as 20%. The final element required to generate costs for each worker to complete subtasks of the overall manufacturing task is the weighting for each cost function variable for each subtask. The weightings defined in Table 6.2 for are again used for consistency with the settings used for the dynamic task planner to generate the initial task plan in Chapter 6.

7.3.6. Setup of the Simulated Work Shift for the Human-Robot Team

Following the setup of the simulated generation of costs for the human and robot workers across a simulated work shift, it is necessary to define the structure of the simulated work shifts for workers. The simulated work shifts to test the task planning methodology are not setup based on an existing work shift for the turbocharger assembly task presented in (Nikolakis et al., 2018), but instead generated to highlight the benefits of task replanning as worker capabilities change across a work shift. In this research, a work shift is defined as consisting of three 1.5 hour work periods for human workers with a half hour rest period between each. This does not represent a realistic work shift structure from an industrial setting, but instead represents a hypothetical situation with exaggerated break periods which allows the human worker's performance to return close to their original levels. This highlights the benefits of task replanning by increasing the visibility of the ability of task replanning to improve the performance of the HR team when a worker's performance improves in addition to when it declines. Although this work shift pattern is used to test the task planning methodology, the task planner can be used in conjunction with a work shift pattern defined by the manufacturer applying this system.

In this research, it is assumed that the human and robot worker work collaboratively during each 1.5 hour work period until a human worker is assigned a break period. During this work period, new sets of task assignments and plans are generated based on the simulated worker performance under the replanning constraints defined in Section 7.2. When the human worker is due a break period, it is assumed that the current task iteration should be

completed with the human worker's break starting from the beginning of the next task iteration. This is reasonable to assume as the initial task completion times given in Chapter 6 indicate that a task iteration should be completed in approximately 2 minutes.

During break periods for the human worker, it is proposed that the robot worker would continue to complete the manufacturing task alone by completing all available subtasks themselves. This is possible for the simulated work shifts since the robot worker is capable of performing all subtasks in the example manufacturing tasks described in Chapter 6, however, it is acknowledged that this may not be possible for all industrial manufacturing tasks. This is considered desirable as it allows production to continue for a HR team's manufacturing cell at a less efficient rate rather than stopping completely during a break period for a human worker. It is once again assumed that when a human worker returns from their break period, that the robot should continue to complete the current task iteration alone with the human worker being reintegrated into the task beginning with the next task iteration. This is reasonable to assume given a task iteration length of a few minutes, as leaving the human worker inactive for a few minutes would be less disruptive than reintegrating a human worker into a partially completed manufacturing task. Once the last active period for the human worker is completed, it is assumed that the current task iteration is completed and then the simulated work shift ends.

As highlighted in Section 7.1, two groups of work shift scenarios are simulated using this work shift structure with the simple and complex example manufacturing tasks from Chapter 6. In the first group of scenarios, the task planner is tested for reaction to changes in performance of the human worker as detected by the continuous cost function variables defined in Chapter 4. Such continuous changes in performance are defined by changes in completion time and the scenarios considered include the human worker performing as expected in addition to scenarios where the human worker is over or under fatigued as described in Section 7.3.2. In these scenarios, it is assumed that the capabilities of the human or robot worker do not change, and they completed any subtasks with an associated error variable within the tolerances described in Section 7.3.4.

In the second group of scenarios, the task planner is tested for reaction to changes in capabilities of the human and robot workers as detected by the discrete cost function variable defined in Chapter 5. To generate this group of scenarios the error scenarios shown in Table 7.2 are applied in subtask 5 as described in Section 7.3.4. This subtask is chosen as the human worker should be assigned this subtask when performing as expected. It is also assumed in these scenarios that the human worker's performance does not change and their completion times are as expected across the simulated work shift. Given

the methodology defined across this section to generate costs for a human and robot worker to complete the simple and complex tasks from Chapter 6 it is possible to determine the effect of task replanning across the simulated work shift.

7.4. Dynamic Task Planner Results: Replanning Sets of Task Assignments and Task Plans Across a Work Shift

7.4.1. Simple Manufacturing Task Across a Work Shift

Before presenting the results for task replanning for the HR team in the simple task across a work shift, it is important to detail an inconsistency in the number of task iterations between task replanning attempts. This was caused by a coding error in the simulations which meant that after a break period replanning occurs after 9 iterations for the first replan during the new work period instead of 10. This should not have a significant effect on the performance of the HR team as the task is still replanned in a structured way which was the reasoning for using a set number of task iterations between task replanning attempts.

Acknowledging this inconsistency, it is first necessary to analyse the dynamic task planner's capabilities when the human worker performs as expected and no errors are made for the simple manufacturing task. For this simulated work shift, Figure 7.3 shows the total cost for the simple task in each task iteration of the simulated work shift for the initial set of task assignments and task plan generated from historic data in addition to when the task is replanned across the work shift. Here Figure 7.4 shows the corresponding total completion times for each task iteration across the simulated work shift and Figure 7.5 shows the percentage of subtasks assigned to each worker across the simulated work shift when the task is replanned. Finally, Figure 7.6 shows the execution time for the dynamic task planner each time that task planning occurs across the simulated work shift.

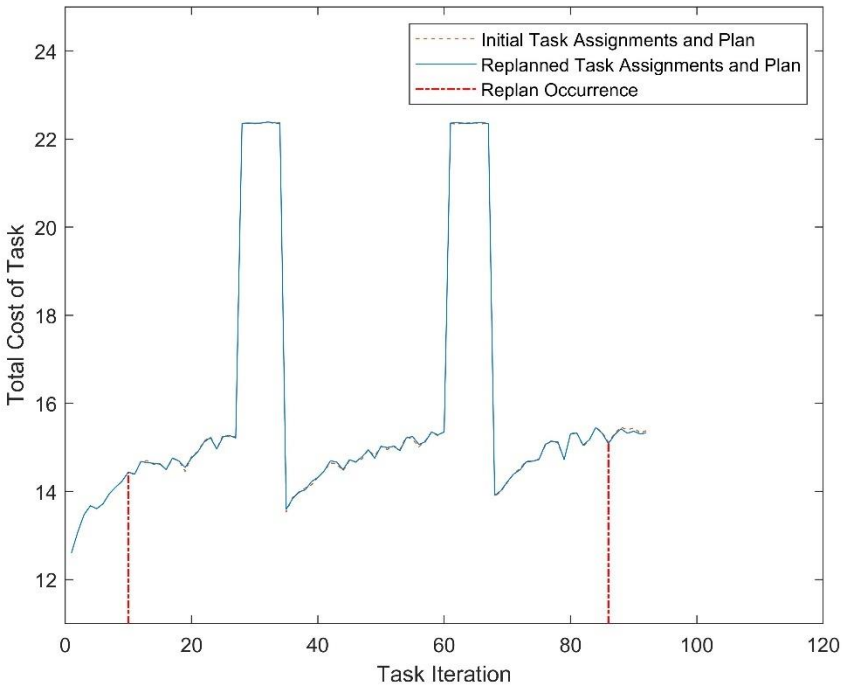

Figure 7.3: A plot of the total cost for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was performing as expected.

Figure 7.3 shows that task replanning is only attempted twice when the simulated human worker is performing as expected, once towards the start of the simulated work shift and once towards the end. It is shown in Figures 7.3 and 7.4 that the first occurrence of replanning has no effect on the total cost or total completion time for the HR team to complete the task across the work shift until the next task replan. This occurs as the task assignments and task plan are not changed by task replanning, indicating that the dynamic task planner could not find a better set of task assignments and task plan for the HR team given the increase in cost over the first 10 iterations of the manufacturing task. Figures 7.3 and 7.4 show that the second attempt at replanning results in a very minimal improvement of the total cost and total completion time for the HR team to complete the simple manufacturing task. This improvement is generated by reassigning the 8^{th} subtask from the human worker to the robot worker with the task plan remaining unchanged.

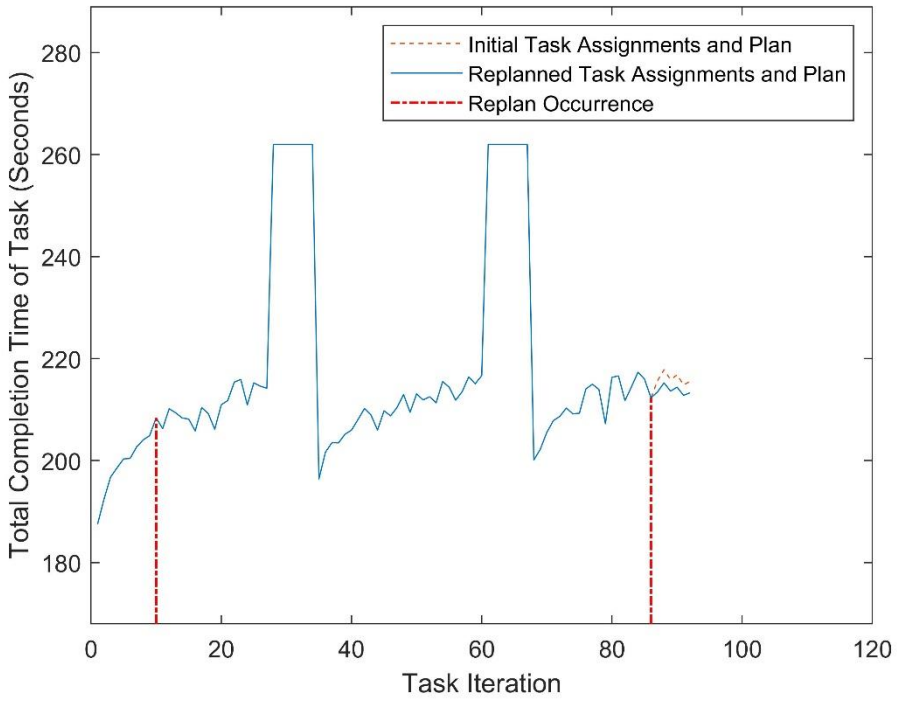

Figure 7.4: A plot of the total completion time for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was performing as expected.

In the case of the simulated human worker performing as expected in the simple manufacturing task it is shown that task planning does not improve the overall performance of the HR team with 92 iterations of the simulated task executed regardless of whether task planning occurs across the simulated work shift. It is shown in Figure 7.6 that the execution time of the task planner more than halves during the work shift from 1.86 seconds to generate the initial task plan to 0.8 seconds and 0.82 seconds, respectively, for each task replan. This behaviour is shown as a greater number of task assignments are “locked in” by the input constraints across a work shift as worker performance deviates, reducing the solution space for the task planner and thus improving its performance.

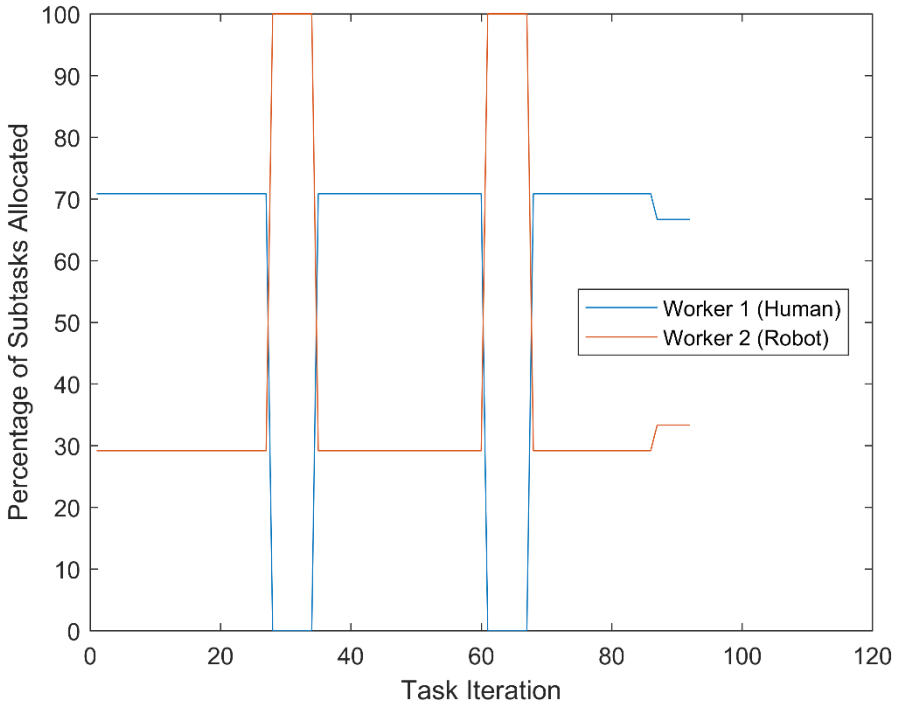

Figure 7.5: A plot of the percentage of subtasks assigned to the simulated human and robot worker across the simulated work shift with the occurrence of task replanning whilst the simulated human worker was performing as expected.

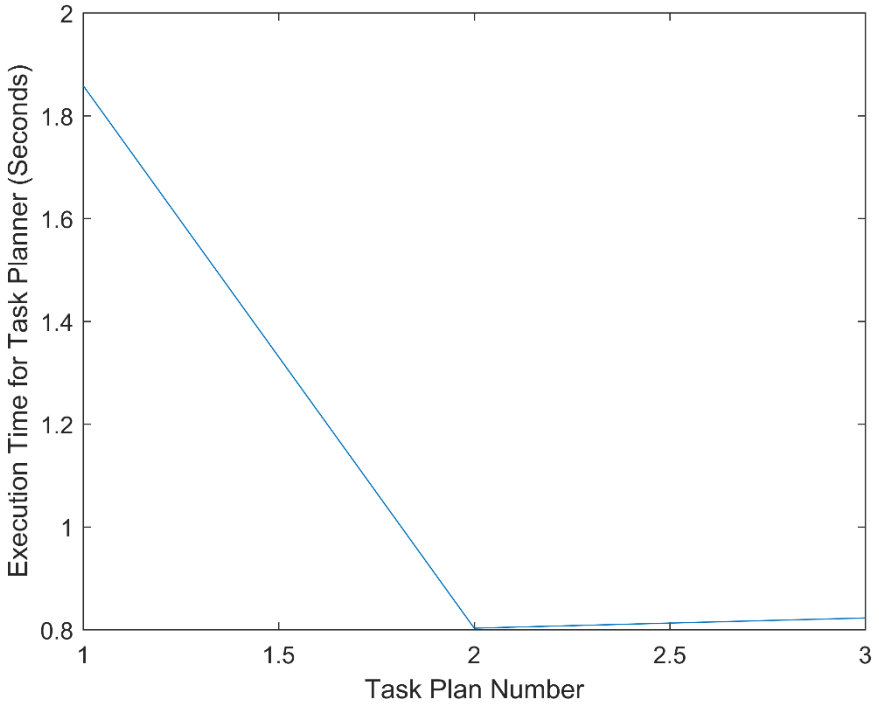

Figure 7.6: A plot of the total execution time for each time the task planner was utilised with the simple task across the simulated work shift whilst the simulated human worker was performing as expected.

For the scenario when the human worker is over fatigued, Figure 7.7 shows the total cost for the simple task in each task iteration of the simulated work shift for the initial set of task assignments and task plan generated from historic data in addition to when the task is replanned across the work shift. Here Figure 7.8 shows the corresponding total completion times for each task iteration across the simulated work shift and Figure 7.9 shows the percentage of subtasks assigned to each worker across the simulated work shift when the task is replanned.

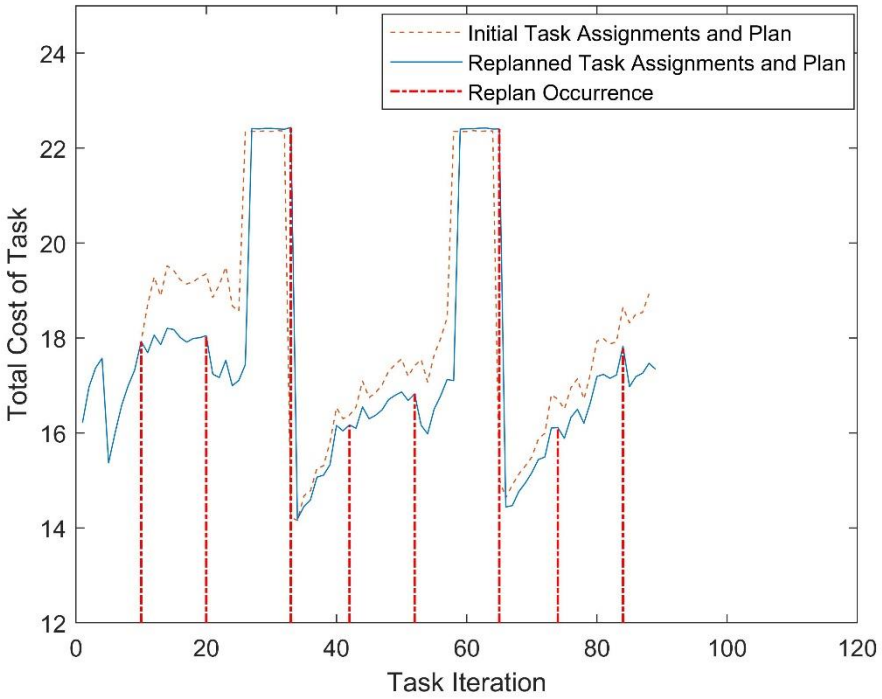

Figure 7.7: A plot of the total cost for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was over fatigued.

Figure 7.7 shows that task replanning is attempted much more frequently in this scenario with eight occurrences, replanning occurs twice during each work period for the human worker in addition to occurrences in the last task iteration before the human worker returns from a break period. In this scenario replanning is shown to cause a noticeable reduction in the total cost for the HR team to complete the simple manufacturing task. The largest reduction in cost occurs between the first replanning occurrence in task iteration 11 and the end of the first work period for the simulated human worker in task iteration 25, with a mean reduction in cost of 1.38. However, the mean reduction in total cost due to replanning is much smaller during the remaining two work periods for the human worker at 0.57 between task iterations 34 and 57 and of 0.69 between task iterations 66 and 88. Figure 7.7 also shows that the HR team is able to complete an additional iteration of the simple manufacturing task during the first two work periods for the human

worker due to the improvement in performance. This necessitates analysis of the mean cost differences over the ranges shown as it would result in an unfair comparison to compare costs with the human and robot competing the task collaboratively to those of the robot completing the task alone which are much higher.

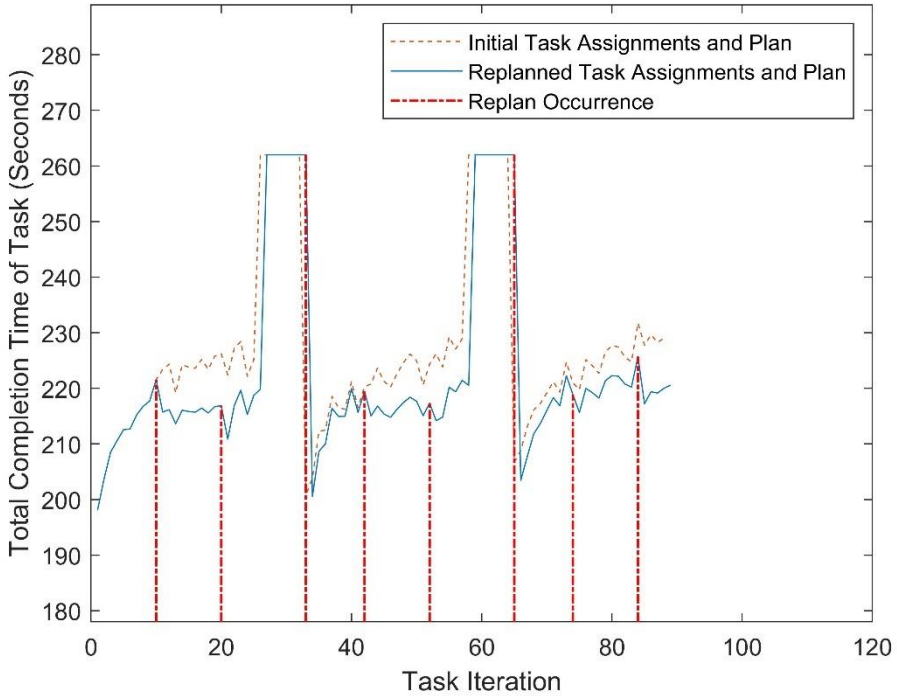

Figure 7.8: A plot of the total completion time for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was over fatigued.

Figure 7.8 shows that the corresponding improvement in the total completion time for the HR team to complete the simple manufacturing task is also much higher in the first work period for the simulated human worker compared to the other two work periods. This is shown by a mean reduction in the total completion time for the HR team of 8.29 seconds between task iterations 11 and 25 compared to a mean reduction of 5.33 seconds between task iterations 34 and 57 and of 5.23 seconds between task iterations 66 and 88. In this scenario it is shown that the reduction in total completion time due to task replanning results in minimal overall improvement in the efficiency of the HR team. This is shown as 89 iterations of the task are completed when replanning occurs in the simulated work shift and 88 iterations of the task are completed when replanning does not occur.

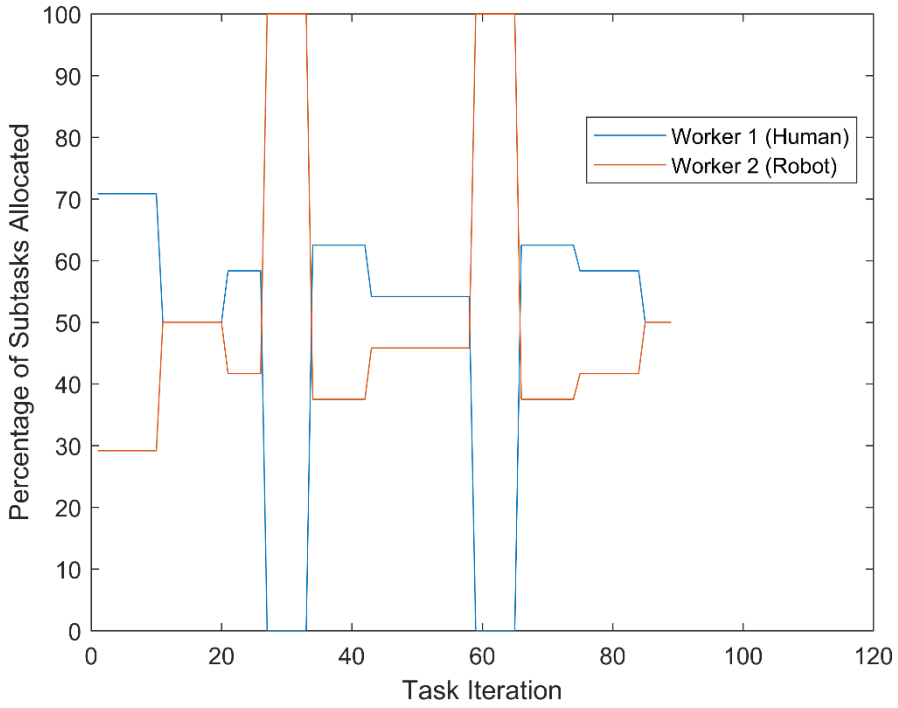

Figure 7.9: A plot of the percentage of subtasks assigned to the simulated human and robot worker in the simple task across the simulated work shift with the occurrence of task replanning whilst the simulated human worker was over fatigued.

Examining the cause of this improvement, Figure 7.9 shows that the dynamic task planner reduces the percentage of subtasks assigned to the human worker across the work shift from the 70.8% used in the initial set of task assignments and task plan. It is also important to first note that the task plan used remains the same across the work shift meaning that the improvement in performance can be attributed to the change in assignment of the subtasks shown in Table 7.3. These task assignment changes do not include those for the human worker's break periods where all tasks are assigned to the robot worker. Figure 7.9 shows that the largest number of subtasks are reassigned with the first occurrence of task replanning, reducing the percentage of subtasks assigned to the human worker from 70.8% to 50%. Table 7.3 shows that this reassigns subtasks 20 and 24 to the robot worker which then remain with the robot for the rest of the work shift. A pattern emerges for the remaining subtasks in Table 7.3 over a work period for a human worker where the worker assigned to these subtasks reverses with each replan. This pattern is shown for replan pairs 1 and 2, 4 and 5 in addition to replans 7 and 8. An exception is shown with replans 3 and 6 which occur when a human worker returns from a break period resulting in subtasks 7, 8, 9 and 22 all being assigned to the human worker instead.

Table 7.3: A table of subtask assignment changes implemented by the dynamic task planner for the HR team completing the simple manufacturing task across the simulated work shift when the simulated human worker was over fatigued.

	Subtasks Reassigned					
Occurrence of Task Planning	**Subtask 7**	**Subtask 8**	**Subtask 9**	**Subtask 20**	**Subtask 22**	**Subtask 24**
Initial	Human	Human	Human	Human	Human	Human
Replan 1	Human	Robot	Robot	Robot	Robot	Robot
Replan 2	Robot	Human	Human	Robot	Human	Robot
Replan 3	Human	Human	Human	Robot	Human	Robot
Replan 4	Human	Robot	Robot	Robot	Human	Robot
Replan 5	Robot	Human	Human	Robot	Robot	Robot
Replan 6	Human	Human	Human	Robot	Human	Robot
Replan 7	Human	Robot	Human	Robot	Human	Robot
Replan 8	Robot	Human	Robot	Robot	Robot	Robot

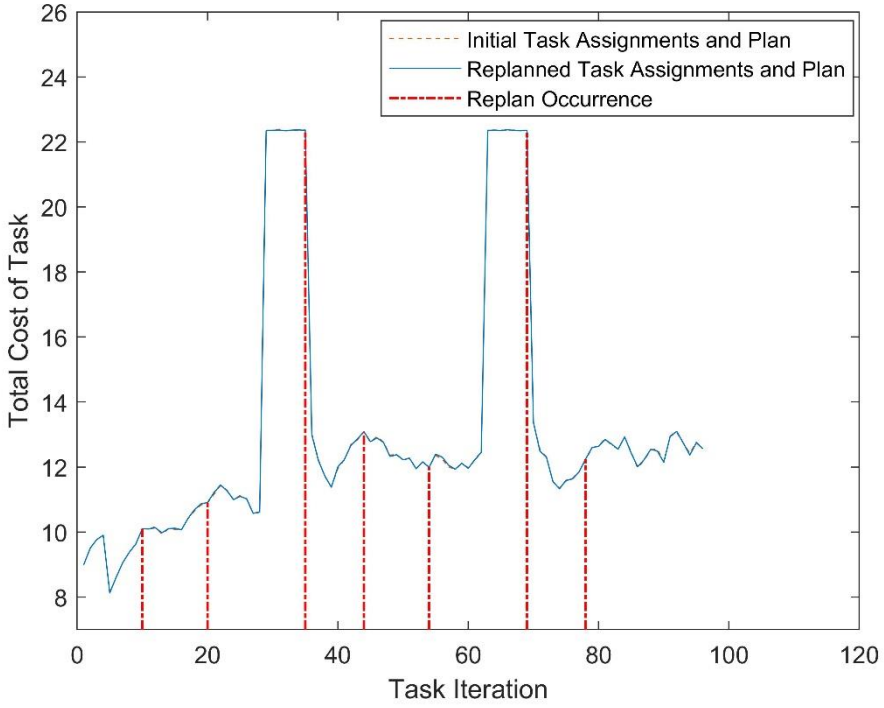

Figure 7.10: A plot of the total cost for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was under fatigued.

Next, it is necessary to analyse the effect of task replanning in the scenario when the human worker is under fatigued. Here, Figure 7.10 shows the total cost for the HR team to complete the simple task in each task iteration of the simulated work shift for the initial set of task assignments and task plan generated from historic data in addition to when the task is replanned across the work shift. Figure 7.10 shows that task replanning has no effect on the total

cost for the HR team to complete the simple manufacturing task in this case as despite 7 attempts at replanning, a better set of task assignments and task plan could not be found. This also results in no overall improvement in the performance of the HR team with 96 iterations of the simple task executed regardless of whether task planning occurs across the simulated work shift. This highlights a limitation of the dynamic task planner in that a human worker cannot be assigned a subtask if they were not assigned the subtask in the initial set of task assignments. It is shown in Figure 7.12 that 70.8% of subtasks are assigned to the human worker across this simulated work shift, assigning any additional subtasks to them would have resulted in the limit of no more than 75% of subtasks being assigned to a single worker being broken.

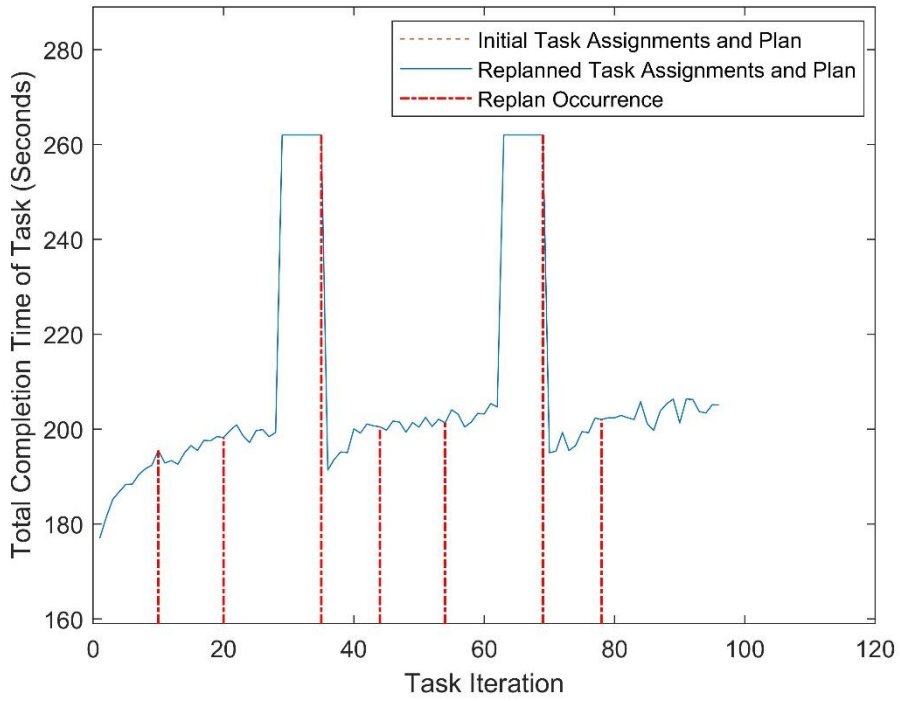

Figure 7.11: A plot of the total completion time for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was under fatigued.

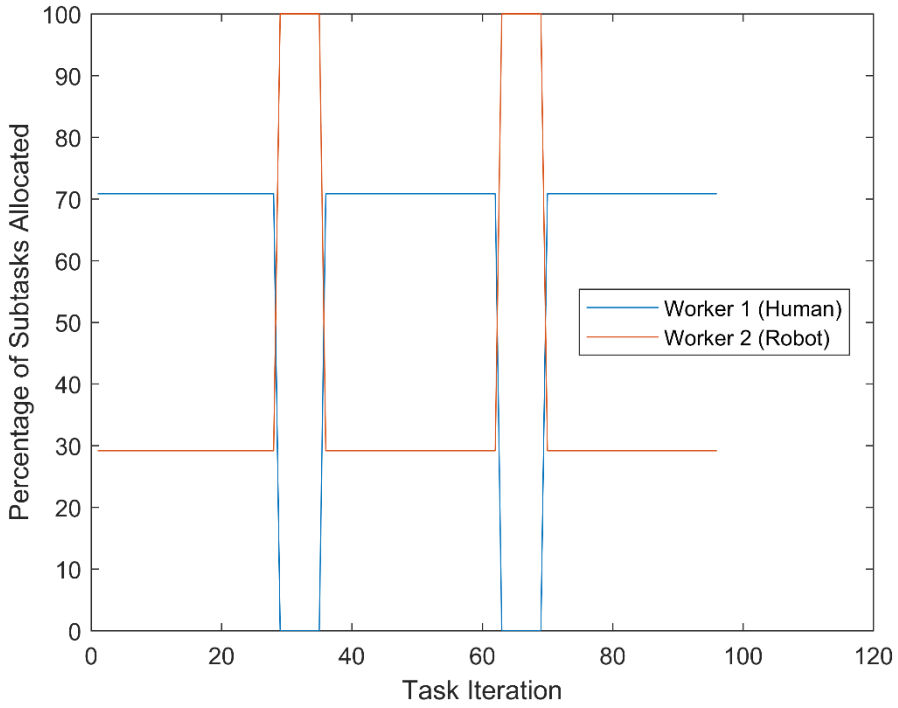

Figure 7.12: A plot of the percentage of subtasks assigned to the simulated human and robot worker across the simulated work shift with the occurrence of task replanning whilst the simulated human worker was under fatigued.

It is next necessary to analyse the capabilities of the task planner when the human has nominal fatigue levels but makes errors completing subtask 5 of the overall manufacturing task. First analysing the case of error scenario 3 occurrence set 1, Figure 7.13 shows the total cost for the HR team across the simulated work shift with and without the occurrence of replanning. In this scenario, the only difference in cost between the initial set of task assignments and task plan compared with when replanning occurs is a mean cost increase of 0.1708 between task iterations 11 and 20. Figure 7.14 shows that the corresponding total completion time to complete the task increases by a mean value of 2.47 seconds between task iterations 11 and 20. In this scenario the HR team completes 92 task iterations with or without the occurrence of replanning across the simulated work shift despite this increase in total completion time.

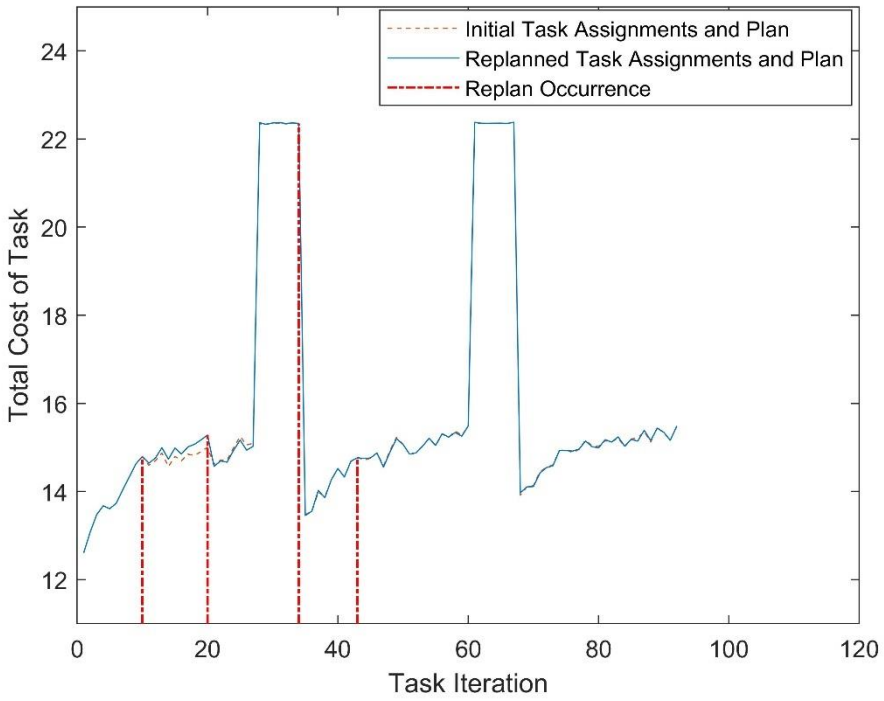

Figure 7.13: A plot of the total cost for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was performing as expected but subject to error scenario 3 occurrence set 1.

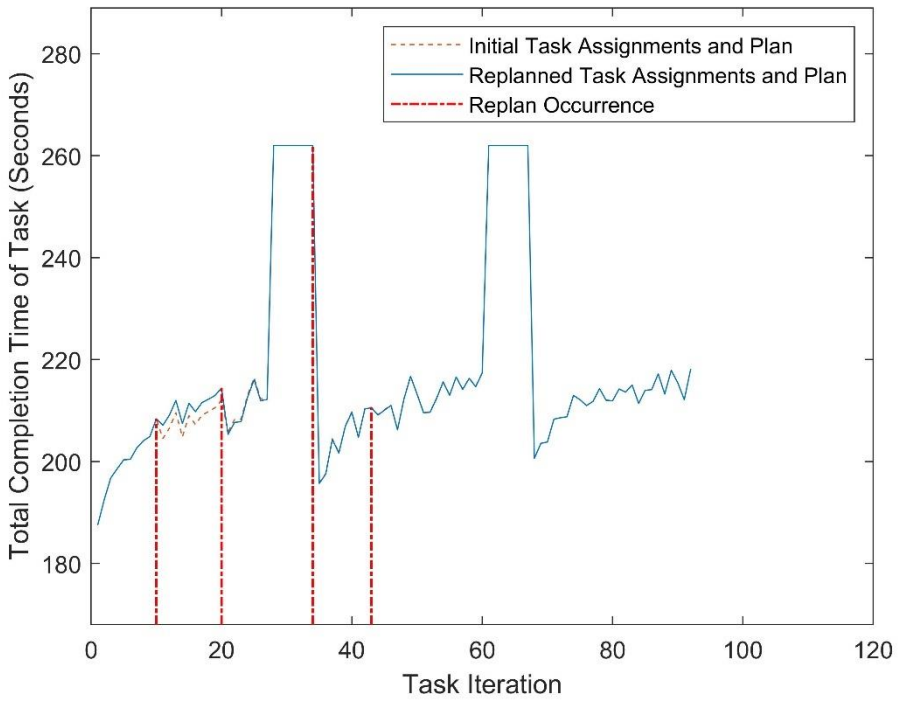

Figure 7.14: A plot of the total completion time for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was performing as expected but subject to error scenario 3 occurrence set 1.

Examining the cause of this behaviour, it is shown that only the assignment of subtask 5 changes over the simulated work shift with the exception of the reassignment of all subtasks to the robot worker in the human worker's break periods as shown in Figure 7.15. Despite this, the task plan for the HR team to complete the simple task does not change in any replanning attempt. In this simulated work shift, the task planner reassigns subtask 5 from the human worker to the robot worker after the first replanning occurrence at the end of task iteration 10. This can be attributed to the increase of the discrete error variable for the human worker in this subtask triggered by the errors induced by error scenario 3 occurrence set 1. Here, the discrete error variable reaches its maximum value in task iteration 9 and its cost is still high enough in subtask iteration 10, despite a successfully completed task iteration, to affect the human worker's total cost to complete that subtask and trigger task reassignment.

It is shown that subtask 5 is then reassigned back to the human worker after the second replanning occurrence at the end of task iteration 20 and remains so for the remainder of the work shift. This occurs as the recovery model is used to generate completion times for the human worker when they are no longer assigned the subtask, reducing them and in turn the cost of the completion variable. This reduces their total cost to complete the subtask, despite the discrete error variable remaining high, allowing the task to be reassigned back to them. Across the remainder of the work shift, the discrete error variable cost for the human worker reduces with successfully completed iterations of the subtask, as it is assumed that the worker completes the subtask within tolerances unless discrete events are triggered. This also explains the deviation in cost between the replanned and initial sets of task assignments and task plans as when the human worker continues to be assigned the subtask it is simulated that they are completing the subtask successfully. This results in the same effect of their cost for the discrete variable being reduced with every successfully completed task iteration.

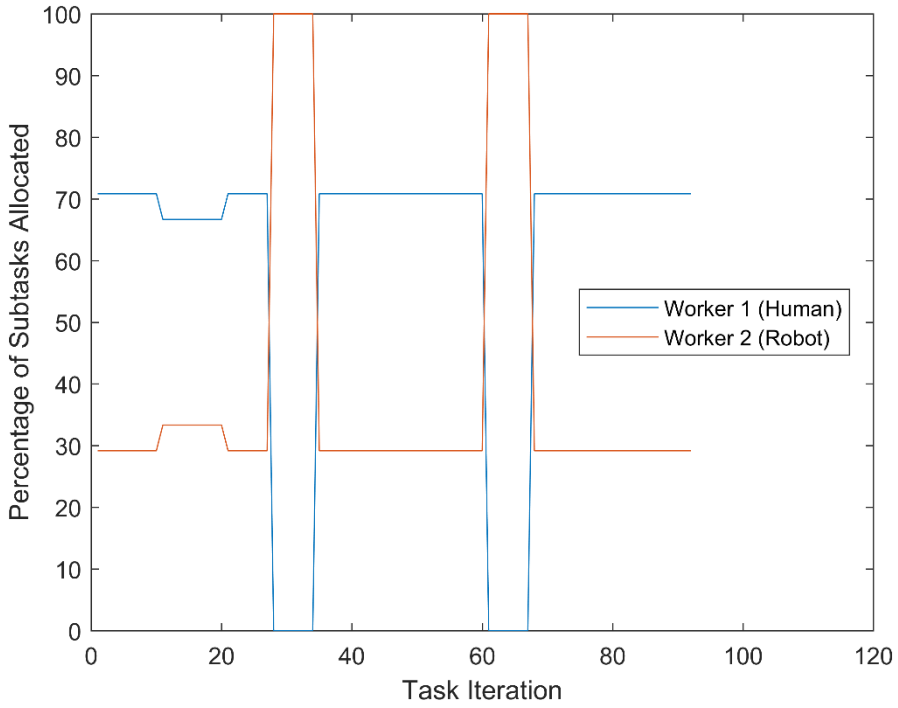

Figure 7.15: A plot of the percentage of subtasks assigned to the simulated human and robot worker across the simulated work shift with the occurrence of task replanning whilst the simulated human worker was performing as expected but subject to error scenario 3 occurrence set 1.

Second, it is necessary to analyse the case of error scenario 3 occurrence set 2 to determine the effect on task replanning of error scenario 3 occurring at a different point in the simulated work shift. Here, Figure 7.16 shows that the total cost for the HR team to complete the simple task for the initial set of task assignments and task plan generated from historic data in addition to when the task is replanned across the work shift. Figure 7.16 shows a very minimal improvement in cost is achieved between the fourth occurrence of replanning and the end of the work shift, with a mean reduction in cost of 0.025 between task iterations 54 and 60 and of 0.035 between task iterations 68 and 92.

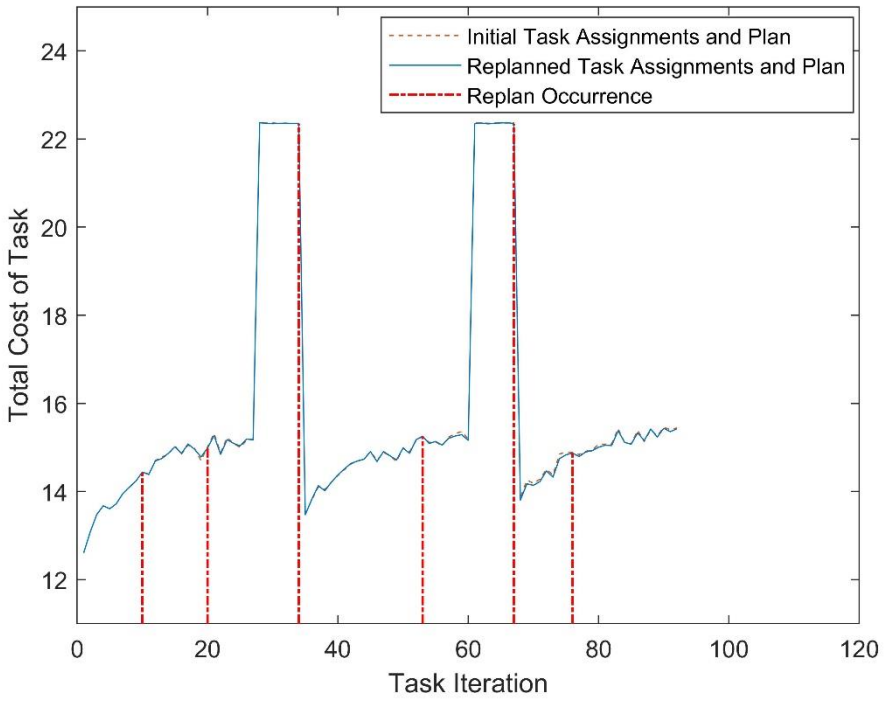

Figure 7.16: A plot of the total cost for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was performing as expected but subject to error scenario 3 occurrence set 2.

Figure 7.17 shows that there is a corresponding improvement in the total completion time for the HR team to complete the simple manufacturing task, however, the magnitude is minimal compared to the total completion time for the task. This is shown by a mean improvement of total completion time of 2 seconds between task iterations 54 and 60 and a mean improvement of 0.39 seconds between task iterations 68 and 92. This minimal reduction in total completion time for the HR team to complete the task does not improve its efficiency as 92 task iterations are completed with or without the occurrence of replanning across the simulated work shift.

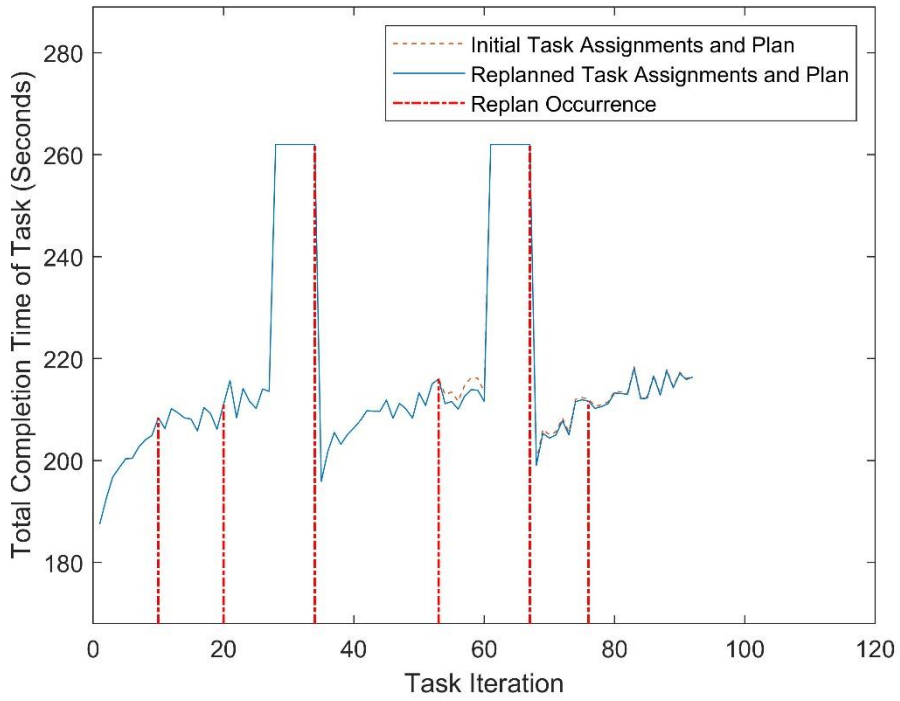

Figure 7.17: A plot of the total completion time for the simulated human and robot workers to complete the simple manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was performing as expected but subject to error scenario 3 occurrence set 2.

Examining the cause of this improvement shows that only the assignment of subtask 8 changes over the simulated work shift with the exception of all subtasks being reassigned to the robot worker during the human worker's break period as shown in Figure 7.18. It is again important to note that throughout this simulated work shift the task plan for the HR team to complete the simple task does not change. Here, the task planner reassigns subtask 8 from the human worker to the robot worker in the fourth occurrence of replanning at the end of task iteration 53, however, the subtask is returned back to them with the fifth occurrence of replanning at the end of their second break period in task iteration 67.

In this scenario it is shown that the task planner does not reassign subtask 5 to the robot worker given this error scenario. In error scenario 3 occurrence set 2, the cost of the discrete precision of sealant application variable reaches its peak value with the final error in sealant application in task iteration 15. As stated previously, unless discrete events are triggered in the simulated work shift then it is assumed that a worker completes the subtask within tolerances. This means that by task iteration 20, the cost of the discrete error variable and thus the total cost for the human worker to complete subtask 5 reduces to the point where it is not reassigned to the robot worker.

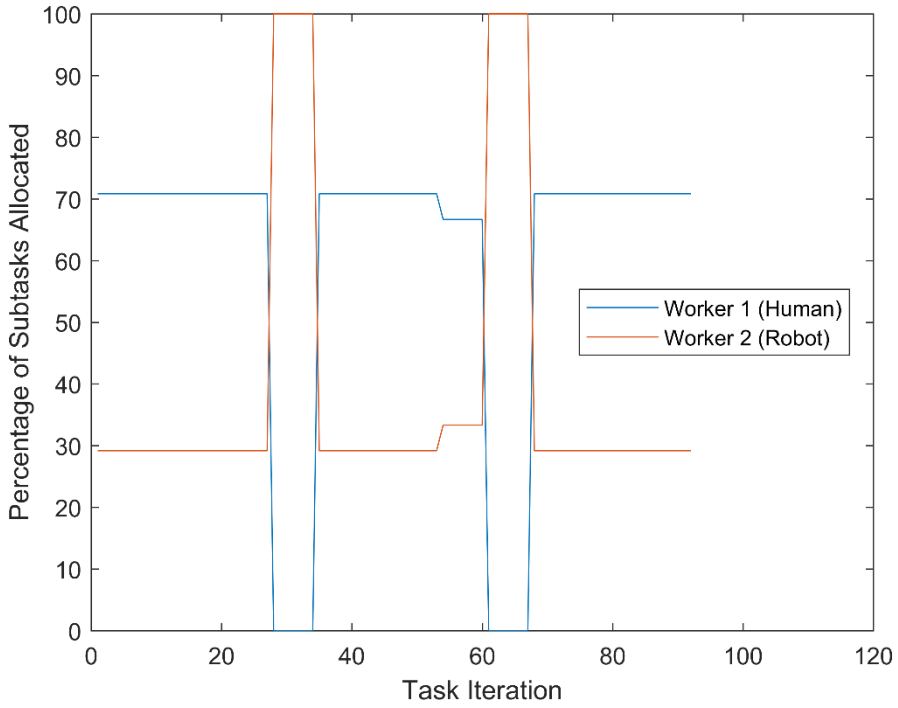

Figure 7.18: A plot of the percentage of subtasks assigned to the simulated human and robot worker across the simulated work shift with the occurrence of task replanning whilst the simulated human worker was performing as expected but subject to error scenario 3 occurrence set 2.

7.4.2. Complex Manufacturing Task across a Work Shift

It is next necessary to simulate these scenarios of worker performance across a work shift for the complex manufacturing task to analyse the capabilities of the full dynamic task planner. First it is necessary to analyse the case when the human worker performs as expected and no errors are made in the complex manufacturing task. Here Figure 7.19 shows the total costs for the HR team to complete the complex task for the initial set of task assignments and task plan generated from historic data in addition to when the task is replanned across the work shift. This shows that task replanning has no effect on the total cost for the HR team to complete the task and that despite six attempts at replanning no better set of task assignments are found. In contrast, the task plan is changed in all replanning occurrences except the sixth replan as shown in Table 7.4, however, this only involves changing the order of completion of the first three subtasks. Despite these changes the HR team completed the same number of 114 task iterations regardless of whether the complex task is replanned or not over the simulated work shift.

Table 7.4: A table of subtask ordering changes implemented by the dynamic task planner for the HR team completing the complex manufacturing task across the simulated work shift when the simulated human worker was performing as expected.

	Subtask Order of Completion in Task Plan		
Occurrence of Task Planning	**1st**	**2nd**	**3rd**
Initial	2	3	1
Replan 1	2	1	3
Replan 2	1	2	3
Replan 3	3	2	1
Replan 4	2	1	3
Replan 5	1	3	2
Replan 6	1	3	2

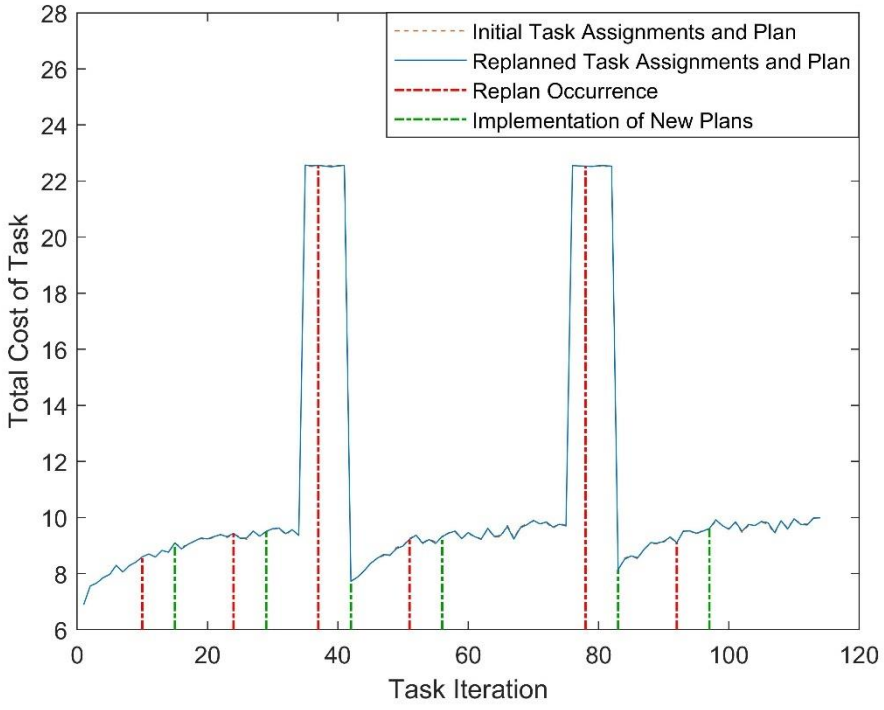

Figure 7.19: A plot of the total cost for the simulated human and robot workers to complete the complex manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was performing as expected.

Figure 7.20 shows that, as with the simple manufacturing task, the execution time of the dynamic task planner significantly decreases from approximately 22.4 minutes to 9.8 minutes from the first task replanning occurrence onwards. This behaviour again occurs as a greater number of task assignments are "locked in" by the input constraints across a work shift as worker performance deviates, reducing the solution space for the task planning problem. Despite this performance improvement, the long execution time of the task planner necessitates that the task is replanned whilst execution

continues, with the new set of task assignments and task plan being implemented when ready. This means that after task replanning is instigated, it takes a mean of five task iterations before the new set of task assignments and task plan are implemented. This gap between the instigation of task replanning and implementing its result risks the relevance of the new set of task assignments and task plan, potentially negating the benefits of task replanning.

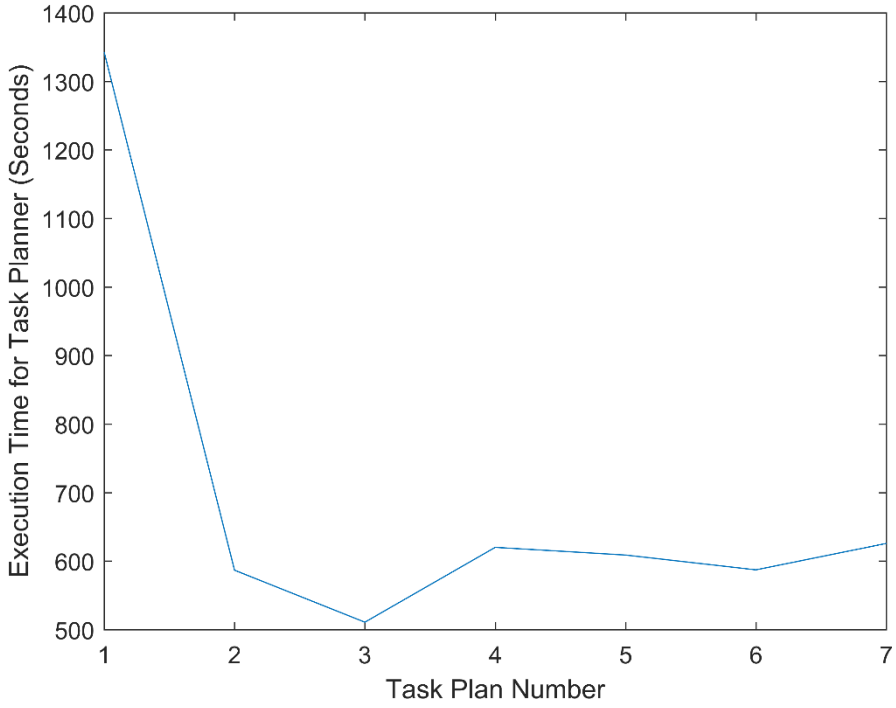

Figure 7.20: A plot of the total execution time for each time the task planner was utilised with the complex task across the simulated work shift whilst the simulated human worker was performing as expected.

For the scenario when the human worker is over fatigued, Figure 7.21 shows the total cost for the HR team to complete the complex task across the simulated work shift with and without the occurrence of task replanning whilst Figure 7.22 shows the corresponding total completion times. This shows task replanning has an inconsistent effect across the simulated work shift with a slight improvement in the total cost for the HR team to complete the complex task in addition to a significant decline in performance. Here Table 7.5 shows the mean cost difference and completion time difference between each set of task assignments and task plans generated by replanning and the initial set of task assignments and task plan generate from historic data. Table 7.5 shows that the first and second occurrences of replanning result in a small mean improvement in cost from those of the initial set of task assignments and task plan. However, these replanning attempts also results in a small mean increase in completion times from those of the initial set of task assignments and task plan. In contrast, Table 7.5 shows that the third occurrence of replanning after

the human worker's first break period results in a very significant mean increase in cost and completion times compared to those of the initial set of task assignments and task plan. Following this, a small mean improvement in cost and a more significant mean improvement in total completion times for the HR team is then seen with the implementation of the fourth replan. However, whilst the fifth task replan maintains the mean improvement in the total completion times for the HR team at a slightly lower magnitude, it instead results in a very small mean increase in cost from those of the initial set of task assignments and task plan. Figure 7.21 shows that despite the improvement in total completion times for the HR team towards the end of the work period, the HR team completes one less task iteration before the human worker starts their second break period compared to when the initial set of task assignments and task plans are used. Finally, it is shown in the last work period that the HR team experiences an insignificant mean cost increase and a small increase in completion times under replanning from those of the initial set of task assignments and task plan between task iterations This changes with the final occurrence of replanning which results in a small mean reduction in cost and completion times for the HR team.

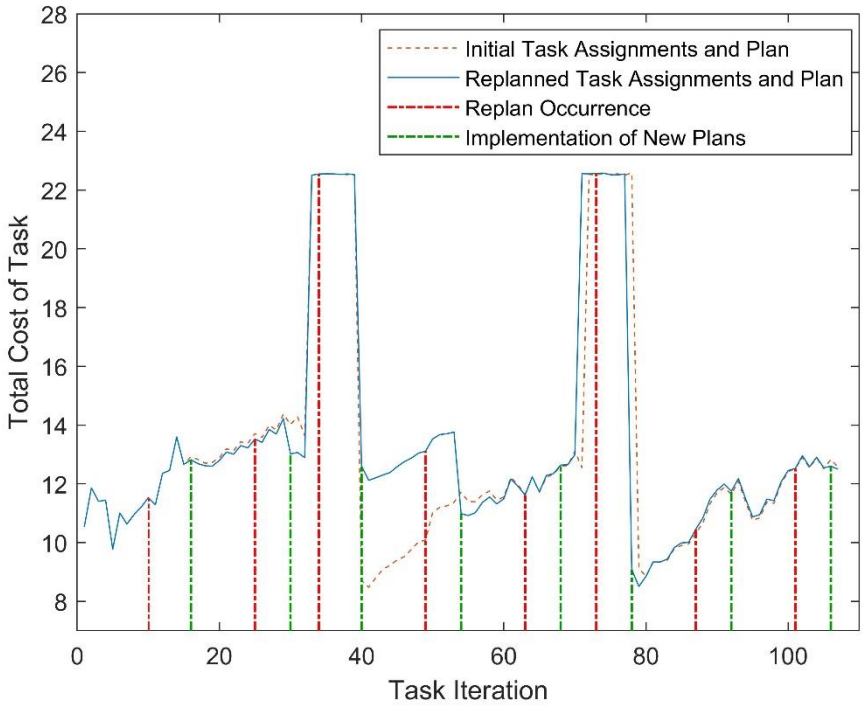

Figure 7.21: A plot of the total cost for the simulated human and robot workers to complete the complex manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was over fatigued.

Table 7.5: A table of the difference in cost and total completion time for the HR team between the replanned task and the initial set of task assignments and task plan.

Occurrence of Task Planning	Task Iterations	Mean Difference in Cost	Mean Difference in Total Completion Time (Seconds)
Replan 1	16 – 29	-0.1323	0.6195
Replan 2	30 – 32	-0.9891	0.3424
Replan 3	40 – 53	3.0506	16.9330
Replan 4	54 – 67	-0.1614	-3.0691
Replan 5	68 – 69	0.0311	-2.6924
Replan 6	79 – 91	0.0225	0.2916
Replan 7	92 – 105	0.0697	0.2147
Replan 8	106 – 107	-0.1517	-0.8704

It is shown over the course of this simulated work shift that the HR team complete one less task iteration with the occurrence of task replanning with 107 task iterations being completed compared to the 108 task iterations completed if the initial set of task assignments and task plan are used. This extra task iteration is not simulated as the number of task iterations simulated is defined by the length of the work shift and the sum of the total completion times across the work shift when the task is replanned.

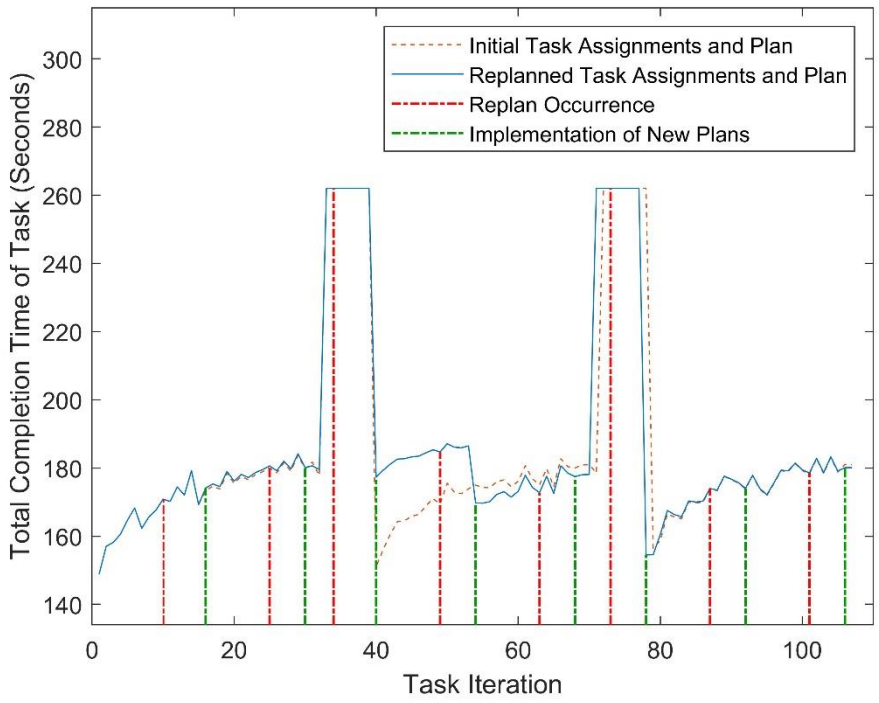

Figure 7.22: A plot of the total completion time for the simulated human and robot workers to complete the complex manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was over fatigued.

Examining the cause of this behaviour, Figure 7.23 shows the percentage of subtasks assigned to each worker across the simulated work shift when the task is replanned whilst Table 7.6 and Table 7.7 show the changes in subtask assignments and task plans, respectively. Figure 7.23 shows an inconsistent reaction in the percentage of subtasks assigned to the human worker in each of the three work periods for the human worker. In the first and third work periods, Table 7.6 shows a decreasing pattern in the percentage of subtasks assigned to the human worker across the work period as shown with the simple manufacturing task. However, Table 7.7 shows that the only corresponding variation in the task plans is the ordering of the first three subtasks with the larger subassemblies in the second half of the task being completed in the same order. In contrast, Table 7.6 shows that the third task replan does not shift the majority of the task assignments back to the human worker following their break period causing the large increase in the cost and completion time seen in Figures 7.21 and 7.22. Additionally, in replan 5 towards the end of this work period, only 3 subtasks are reassigned to the robot worker instead of the 8 shown at the end of the other two work periods. A large variability is also shown in the task plans over this work period with changes in the order of completion of the first three subtasks in addition to the larger subassemblies in the second half of the task with each task replan.

Table 7.6: A table of subtask assignment changes implemented by the dynamic task planner for the HR team completing the complex manufacturing task across the simulated work shift when the simulated human worker was over fatigued. Here a green box labelled H indicates that the human is assigned the subtask whereas a blue box labelled R indicates that the robot is assigned the subtask.

	Subtasks Reassigned												
Occurrence of Task Planning	**5**	**7**	**8**	**10**	**13**	**14**	**16**	**19**	**20**	**21**	**22**	**23**	**24**
Initial	H	H	H	H	H	H	H	H	H	H	H	H	H
Replan 1	H	H	H	H	H	H	H	H	H	H	H	H	R
Replan 2	R	R	H	R	R	H	H	R	R	R	R	H	R
Replan 3	H	H	R	H	H	H	R	H	R	R	H	R	R
Replan 4	H	H	H	H	H	H	H	H	H	H	H	H	R
Replan 5	H	H	H	H	H	R	H	H	H	H	R	R	R
Replan 6	H	H	H	H	H	H	H	H	H	H	H	H	R
Replan 7	H	H	H	H	H	H	H	H	H	H	H	H	R
Replan 8	H	R	H	R	R	H	H	R	R	R	R	R	R

Table 7.7: A table of subtask ordering changes implemented by the dynamic task planner for the HR team completing the complex manufacturing task across the simulated work shift when the simulated human worker was over fatigued.

	Subtask/Subassembly Beginning with Subtask Order of Completion in Task Plan								
Occurrence of Task Planning	**1st**	**2nd**	**3rd**	**4th (Static)**	**5th**	**6th**	**7th**	**8th**	**9th (Static)**
Initial	2	3	1	4	23	18	5	10	24
Replan 1	3	1	2	4	23	18	5	10	24
Replan 2	3	2	1	4	23	18	5	10	24
Replan 3	2	1	3	4	23	5	10	18	24
Replan 4	3	1	2	4	23	18	5	10	24
Replan 5	1	2	3	4	23	5	10	18	24
Replan 6	3	2	1	4	23	18	5	10	24
Replan 7	1	3	2	4	23	18	5	10	24
Replan 8	3	2	1	4	18	23	5	10	24

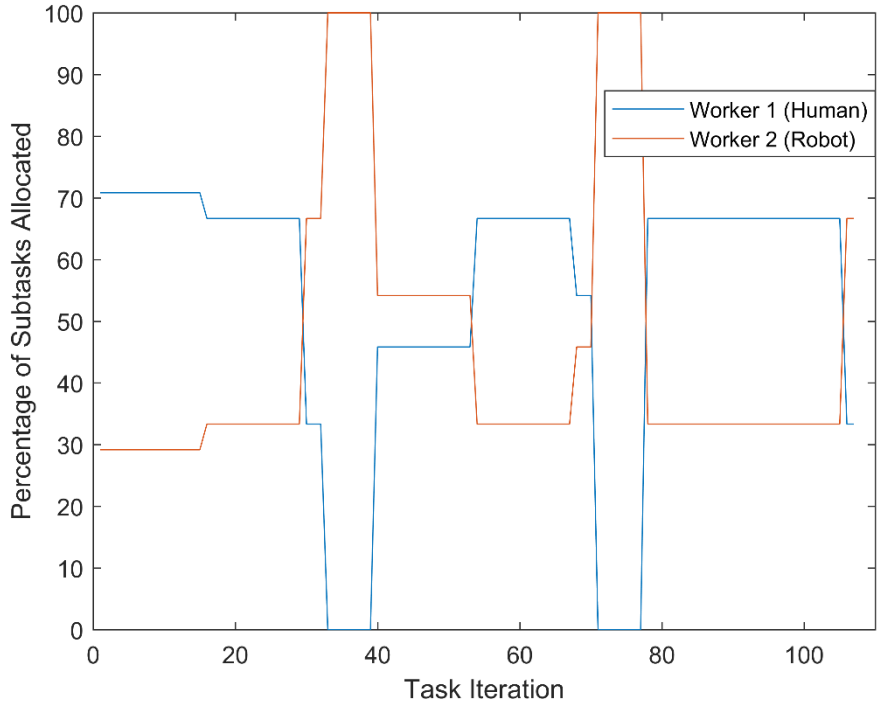

Figure 7.23: A plot of the percentage of subtasks assigned to the simulated human and robot worker in the complex task across the simulated work shift with the occurrence of task replanning whilst the simulated human worker was over fatigued.

For the scenario when the human worker is under fatigued, Figure 7.24 shows the total cost for the HR team to complete complex task in each task iteration of the simulated work shift with and without the occurrence of task replanning. This again shows that task replanning has no effect on the total cost for the HR team to complete the task and despite six attempts at replanning no

better set of task assignments are found. The only variation in the corresponding task plans is the order that the initial 3 subtasks are completed in.

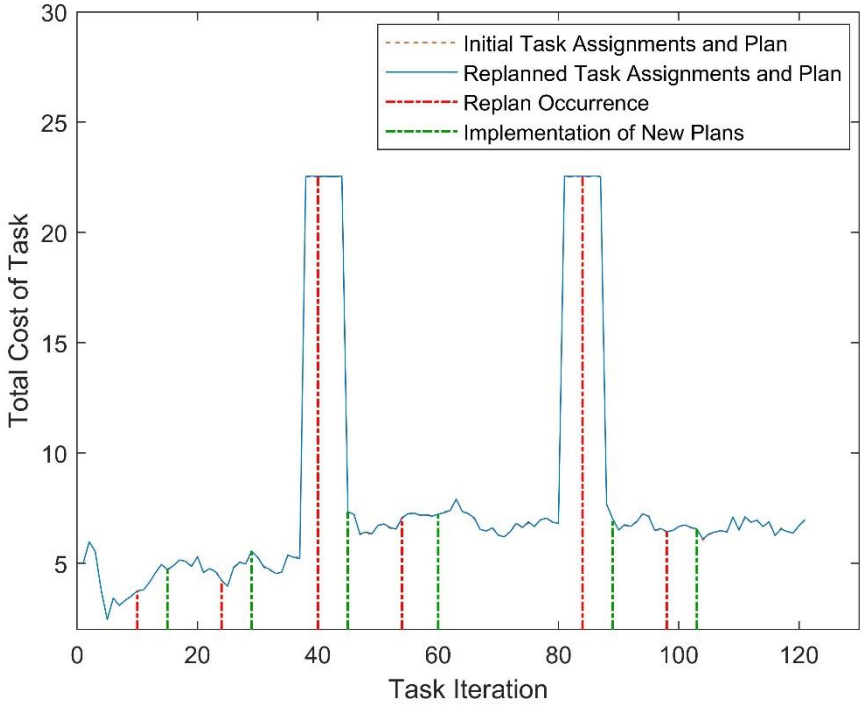

Figure 7.24: A plot of the total cost for the simulated human and robot workers to complete the complex manufacturing task across the work shift, with and without the occurrence of task replanning, whilst the simulated human worker was under fatigued.

Finally, in the scenarios where the human worker is performing as expected but subject to error scenario 3 in subtask 5, task replanning is shown to have no effect on the total cost for the HR team to complete the task in both occurrence sets 1 and 2 of error scenario 3. This is shown despite 7 and 6 occurrences of replanning, respectively, for error scenario 3 occurrence sets 1 and occurrence set 2. In both of these scenarios, the dynamic task planner cannot find a better set of task assignments than the initial set generated from historic data. Additionally, the only variation in the corresponding task plans across the simulated work shifts are again the order that the initial 3 subtasks are completed in. This means that the dynamic task planner does not reassign subtask 5 to the robot worker with the build-up of cost for the human worker due to the errors given by error scenario 3 with either error occurrence set 1 or 2.

7.5. Chapter Summary

Over this chapter, the effect of utilising the dynamic task planner proposed in Chapter 6 was analysed to determine the effect of task replanning on the efficiency of a HR team over a simulated work shift. This involved

applying the dynamic task planner to the simple manufacturing task, utilising only DGSA Layer 1, and the complex manufacturing task, utilising the full dynamic task planner, over several scenarios of human performance. These scenarios were used to determine how the dynamic task planner would react to changes in the performance of the human worker in addition to the occurrence of discrete errors as they are completing the manufacturing tasks alongside the robot worker.

First, in the simple task it was shown that the dynamic task planner could not find a better set of task assignments, or task plan, over the majority of the simulated work shift when the human worker was performing as expected and no errors occurred. The dynamic task planner did find a better set of task assignments towards the end of the work shift by reassigning a subtask to the robot worker. However, this minimally reduced the total cost and completion time for the HR team compared to the initial set of task assignments and task plan. It was shown that the dynamic task planner's execution time reduced from 1.8 s to generate the initial set of task assignments and task plan to approximately 0.8 s during the simulated work shift, as more task assignments were "locked in" due to worker performance. This demonstrates the utility of the dynamic task planner when only DGSA Layer 1 was applied as it enables task assignments to rapidly be revaluated between task iterations to ensure they reflect the current capabilities of the human and robot workers.

The benefits of task replanning were clearly shown when the simulated human worker was over fatigued with a consistent improvement in cost and task completion time with each task replan compared to the initial set of task assignments and task plan. Despite this, the HR team only completed one additional task iteration with the occurrence of task replanning compared to persistent use of the initial set of task assignments and task plan. The dynamic task planner achieved this by assigning more subtasks to the human worker towards the start of their work period when their performance would be better, then transferring more subtasks to the robot worker as their performance declines across a work period. In comparison it was shown that the dynamic task planner could not improve the performance of the HR team when the simulated human worker was under fatigued. This was due to the limitations of the dynamic task planner which prevented subtasks being reassigned to the human worker if they were not assigned them in the first task iteration of the work shift. This decision was made due to the lack of available data on the human worker's current capabilities in these subtasks. In order, to improve the capabilities of the task planner when a human worker is under fatigued it would be necessary to allow these subtasks to be reassigned to human workers.

It was shown that the dynamic task planner was able to react to a build-up in errors and cost of a discrete error variable, that increased the cost for a human worker to complete a subtask. However, this strategy was only successful when the peak cost was achieved in a task iteration close to when the task was replanned. In comparison, the dynamic task planner failed to react to a build-up of errors when the peak cost (and last error) was achieved in a task iteration far from when the task was replanned. This occurred as successfully completed iterations of the subtask subsequent to the peak cost reduced the cost to a level where the next replanning attempt would not be executed. An additional issue also occurred when a subtask was reassigned to a robot worker due to a build-up of errors for the human worker as the recovery model reduced the completion times for the human worker when they were not assigned the subtask. This resulted in the human worker's cost to complete the subtask reducing to the point where the subtask was reassigned back to them regardless of the high cost of the discrete error variable.

Second, for the complex manufacturing task it was shown that the effectiveness of the dynamic task planner was significantly reduced. Here, the dynamic task planner could not find a better set of task assignments than those generated initially when the simulated worker was performing as expected or was under fatigued. Additionally, use of the dynamic task planner resulted in minor changes in the task plan which could be considered unnecessary as they did not improve the efficiency of the HR team. This behaviour was also shown with both error scenarios tested where the dynamic task planner failed to reassign the subtask with the occurrence of errors from the human worker in both cases. Inconsistent performance of the dynamic task planner was shown with the over fatigued simulated human worker, as replanning both improved and reduced the performance of the HR team across the simulated work shift. This resulted in a minor overall reduction in efficiency for the HR team when compared to consistent use of the initial set of task assignments and task plan. This may be attributed to the length of time taken to execute the dynamic task planner in these situations which necessitated the task being replanned whilst the HR team continue to execute the manufacturing task. This resulted in a new set of task assignments and task plan being implemented several iterations after task planning was initiated. This meant that the task assignments and task plans may not represent the current worker capabilities as well as with the immediate implementation shown with the simple manufacturing task. This was also shown with the human worker's break periods where the task planner had to be executed far in advance of the end of the break period in order for a new set of task assignments and task plan to be ready when the human worker returned. Due to the limitations of the simulation with completion times for workers being generated with each task iteration, this meant the completion times of the human worker did not reflect the lower completion times that

would be expected with a sufficient break period. This could be remedied by instantaneously modelling the reduction in human completion times given the length of the break period. However, the dynamic task planner would still not be able to react to any changes in the robot worker's capabilities once task replanning had been initiated.

Overall, these simulations showed that the dynamic task planner could prove to be effective for HR teams where task assignments can be reassigned with a few potential task plans dictating the order of task completion. In comparison, using the full dynamic task planner with much more reconfigurable tasks meant that sets of task assignments and task plans could not be recalculated as quickly, possibly hindering its effectiveness. As a result of this, the dynamic task planner utilising only DGSA Layer 1 represents a much more viable method for replanning tasks for HR teams across a work shift in the short term. The full dynamic task planner still has the potential for the same effectiveness, however, to achieve this further software engineering would be required to allow the dynamic task planner to be executed faster. Another possibility is to limit the application of the full dynamic task planner to manufacturing tasks consisting of subtasks with larger completion times, i.e. subtasks that take longer than 20 to 30 seconds. In such situations there would be a smaller number of task iterations between task replanning being initiated and a new set of task assignments and plans being ready for implementation. This would help to ensure that sets of task assignments and task plans do not loose relevance when they are ready for implementation.

8. Conclusions

8.1. Overall Conclusions

The aim of the research presented in this thesis is to develop a methodology to optimise the implementation of both human and robot workers in a Human-Robot (HR) team whilst allowing adaptability. The literature review in Chapter 2 revealed a knowledge gap for a generalised semi-online task planning methodology for HR teams that updates knowledge on worker capabilities using online data, then use this to plan entire tasks offline to ensure optimisation. To bridge this knowledge gap, a system architecture was proposed for a task planning system to optimise the implementation of HR teams across a work shift. The focus of this research was to develop the core technologies required for the architecture to function to verify the utility of the generalised semi-online task planning approach.

First, dynamic cost functions were developed, consisting of continuous and discrete variables, to assess worker capabilities using online production data. Example continuous and discrete variables were developed that fulfil the research objective to use continuous or discrete data to quantify the capability of workers to complete manufacturing subtasks. This was achieved as these example variables produced a clear distinction in costs for a robot worker and a human worker with varying capabilities. These variables also fulfil the research objective to develop mechanisms for updating the output cost of a cost function variable given online data obtained over iterations of a manufacturing task to ensure their accuracy. This was achieved by using the most recent data for a worker to define the variable cost, with the continuous variables using the latest completion times and the discrete variable assessing the last sealant application. Importantly the dynamic cost functions offer expandability with the ability to add further variables monitoring other aspects of worker capabilities important to manufacturers implementing the system.

Second, a dual-layer dynamic task planner was developed to replan manufacturing tasks across a work shift given worker costs generated form the dynamic cost functions. Testing the dynamic task planner using DGSA Layer 1 only for a more linear manufacturing task, such as those seen currently in industry, showed that it could find good but not optimal solutions in an execution time of a few seconds. This allowed the dynamic task planner to be implemented between task iterations allowing quick reconfiguration of a HR team during a work shift. In contrast testing the full dynamic task planner for a more reconfigurable manufacturing task showed it could find solutions much closer to optimal solutions. However, its execution time requires it to be run in the background whilst the task continues to be completed. This did not fulfil the research objective of the task planning methodology finding an optimum

set of task assignments and task plan for a HR team. However, the dynamic task planner did find good solutions for a HR team given costs generated for each subtask whilst respecting task constraints and minimising worker idle times.

Finally, intelligent methodologies were developed to implement this dynamic task planner across a work shift. This included pre-execution constraints to preassign subtasks to optimal workers if the cost difference between workers exceeded a defined threshold. Additionally, a methodology was developed to replan tasks at set intervals with separate implementations based on the execution time of the dynamic task planner. This was combined with a utility checking function to trigger task replanning if the optimal worker for any subtask changed or if change in worker costs exceeded a defined threshold since the last replanning attempt. These methodologies fulfil the research objectives to implement mechanisms to ensure subtasks are assigned to optimal workers if there is a significant difference in worker capabilities and trigger task replanning at appropriate intervals but only if worker costs changes indicate this is necessary. Importantly, these methodologies allowed the dynamic task planner to improve the HR team's performance across a work shift with changing worker capabilities compared to without the use of task replanning. The remainder of this chapter highlights contributions to knowledge in Section 8.2 resulting from research described in this thesis. Next, Section 8.3 details possible areas of future research to develop methodologies to improve the utility, accuracy, or speed of the task planning system.

8.2. Contributions to Knowledge

To highlight the utility of the research presented in this thesis, it is next necessary to define the resulting contributions to knowledge. The first contribution to knowledge is the development of dynamic cost functions that utilise online production data to update knowledge on worker capabilities across a work shift. This was achieved through continuous variables that react to gradual changes in worker performance by analysing continuous online production data. Additionally, a discrete variable was developed to react to instantaneous changes in worker capabilities identified by discrete events in manufacturing subtasks. These dynamic cost functions build upon previous approaches of using cost functions to assess worker capabilities, however, the introduction of these continuous and discrete variables for the first time ever provides a way to update costs as capabilities change across a work shift.

The second contribution to knowledge is the development of a dual-layer task planning algorithm to search both task assignments and task plans for a HR team. This represents a generalised algorithm that allows easy encoding of task information and adaption to different manufacturing tasks. Unlike previous methodologies, the task planning algorithm can be adapted to

suit different compositions of HR teams containing various numbers of human and robot workers whilst optimising the task as a whole.

The third contribution to knowledge is the development of a methodology to implement task replanning across a work shift. This utilises intelligent methods to trigger when replanning attempts occur and determine if task replanning is necessary to avoid unnecessary computational expense. Additional methodologies for this are implemented in preconditions of the task planning algorithm itself, such as “locked in” subtasks to ensure a worker is assigned a subtask if there is a sufficient cost difference between them and other workers. This methodology of implementing task replanning across a work shift builds upon state-of-the-art approaches to update task assignments and plans during a work shift. However, the use of pre-execution constraints and dynamic cost functions for the first time ever allows tasks assignments and plans to be updated based on online knowledge of workers performance obtained from the collaborative workspace.

8.3. Future Work

8.3.1. Development of Long-Term Worker Predication

To further enhance the capabilities of task planning, another possible research avenue is to develop long-term prediction of worker capabilities through analysis of historic worker production data and its change across work shifts with task replanning. This approach would allow the development of methods to enhance the task replanning procedure by predicting how to change task assignments and plans based on indicators of change in online production data. Utilising such methods could also allow evolution of the methodology for triggering task planning attempts by utilising indicators of change in online production data to trigger when replanning should occur.

8.3.2. Further Development of Methodologies for Instigating Task Replanning

To allow the dynamic task planner to better react to a build-up of errors by a worker, methodologies for instigating replanning attempts must be further developed. Whilst task replanning at set intervals allowed reaction to gradual changes in workers performance, Chapter 7 showed that this resulted in error build ups being missed if the peak worker cost did not occur close enough to a replanning attempt. To avoid such situations, it is instead necessary to check for a significant increase in worker costs to complete subtasks and implement replanning immediately when this is detected. However, it would also be necessary to define when task replanning should next occur after this to ensure that there is a suitable interval between task replanning attempts.

8.3.3. Improving the Efficiency of The Dynamic Task Planner

Testing of the full dynamic task planner in Chapter 7 highlighted its large execution time as a cause of inconsistent performance. Multiple avenues can be explored to reduce the dynamic task planner's execution time and improve its performance. One possibility is to use parallel processing to implement DGSA Layer 2 for each searcher agent in DGSA Layer 1 as these can be evaluated independently. Additionally, further software development such as the streamlining of the code implementing the dynamic task planner could be used to improve its efficiency.

References

Aaltonen, I., Salmi, T. and Marstio, I. (2018) “Refining levels of collaboration to support the design and evaluation of human-robot interaction in the manufacturing industry,” *Procedia CIRP*, 72, pp. 93–98.

ABB (2020) *YuMi® - IRB 14000 | Collaborative Robot*. Available at: https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi (Accessed: April 3, 2020).

Asfour, T., Kaul, L., Wächter, M., Ottenhaus, S., Weiner, P., Rader, S., Grimm, R., Zhou, Y., Grotz, M., Paus, F. and others (2018) “Armar-6: A collaborative humanoid robot for industrial environments,” *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*., pp. 447–454.

Bdiwi, M., Pfeifer, M. and Sterzing, A. (2017) “A new strategy for ensuring human safety during various levels of interaction with industrial robots,” *CIRP Annals*, 66(1) Elsevier, pp. 453–456.

Blum, C. and Roli, A. (2003) “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM computing surveys (CSUR)*, 35(3) Acm New York, NY, USA, pp. 268–308.

Bonneville, F., Perrard, C. and Henrioud, J.M. (1995) “A genetic algorithm to generate and evaluate assembly plans,” *Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA’95*., Vol.2, pp. 231–239.

Bruno, G. and Antonelli, D. (2018) “Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells,” *The International Journal of Advanced Manufacturing Technology*, 98(9–12) Springer, pp. 2415–2427.

Busch, B., Cotugno, G., Khoramshahi, M., Skaltsas, G., Turchi, D., Urbano, L., Wächter, M., Zhou, Y., Asfour, T. and Deacon, G. (2019) “Evaluation of an industrial robotic assistant in an ecological environment,” *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, pp. 1–8.

Cao, T. and Sanderson, A.C. (1998) “AND/OR net representation for robotic task sequence planning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(2) IEEE, pp. 204–218.

Casalino, A., Mazzocca, E., di Giorgio, M.G., Zanchettin, A.M. and Rocco, P. (2019a) “Task scheduling for human-robot collaboration with uncertain duration of tasks: a fuzzy approach,” *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*., pp. 90–97.

Casalino, A., Zanchettin, A.M., Piroddi, L. and Rocco, P. (2019b) “Optimal Scheduling of Human-Robot Collaborative Assembly Operations With Time Petri Nets,” *IEEE Transactions on Automation Science and Engineering*, IEEE

Chan, W.P., Nagahama, K., Yaguchi, H., Kakiuchi, Y., Okada, K. and Inaba, M. (2015) “Implementation of a framework for learning handover grasp configurations through observation during human-robot object handovers,” *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, pp. 1115–1120.

Colgate, J.E., Wannasuphoprasit, W. and Peshkin, M.A. (1996) “Cobots: robots for collaboration with human operators,” *American Society of Mechanical Engineers, Dynamic Systems and Control Division (Publication) DSC*. ASME, Vol.58, pp. 433–439.

Cremer, S., Mastromoro, L. and Popa, D.O. (2016) “On the performance of the Baxter research robot,” *2016 IEEE international symposium on assembly and manufacturing (ISAM)*., pp. 106–111.

Daoud, S., Chehade, H., Yalaoui, F. and Amodeo, L. (2014) “Solving a robotic assembly line balancing problem using efficient hybrid methods,” *Journal of Heuristics*, 20(3) Springer, pp. 235–259.

Dawson, D., Noy, Y.I., Härmä, M., Åkerstedt, T. and Belenky, G. (2011) “Modelling fatigue and the use of fatigue models in work settings,” *Accident Analysis & Prevention*, 43(2), pp. 549–564.

Digiesi, S., Kock, A.A.A., Mummolo, G. and Rooda, J.E. (2009) “The effect of dynamic worker behavior on flow line performance,” *International Journal of Production Economics*, 120(2), pp. 368–377.

Dowlatshahi, M.B., Nezamabadi-Pour, H. and Mashinchi, M. (2014) “A discrete gravitational search algorithm for solving combinatorial optimization problems,” *Information Sciences*, 258, pp. 94–107.

Dziki, K. and Krenczyk, D. (2019) “Mixed-model assembly line balancing problem with tasks assignment,” *IOP Conference Series: Materials Science and Engineering*., Vol.591, p. 12013.

Erol, K., Hendler, J.A. and Nau, D.S. (1994) UMCP: A Sound and Complete Procedure for Hierarchical Task-network Planning. *Aips*.

Georgievski, I. and Aiello, M. (2015) “HTN planning: Overview, comparison, and beyond,” *Artificial Intelligence*, 222 Elsevier, pp. 124–156.

Ghallab, M., Nau, D. and Traverso, P. (2004) *Automated Planning: theory and practice*. Elsevier.

Glock, C.H., Grosse, E.H., Kim, T., Neumann, W.P. and Sobhani, A. (2019) “An integrated cost and worker fatigue evaluation model of a packaging process,” *International Journal of Production Economics*, 207, pp. 107–124.

Goto, H., Miura, J. and Sugiyama, J. (2013) “Human-robot collaborative assembly by on-line human action recognition based on an fsm task model,” *Human-Robot Interaction 2013 Workshop on Collaborative Manipulation*.

Grigore, E.C., Roncone, A., Mangin, O. and Scassellati, B. (2018) “Preference-based assistance prediction for human-robot collaboration tasks,” *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*., pp. 4441–4448.

Gu, T., Bahri, P.A. and Cai, G. (2003) “Timed Petri-net based formulation and an algorithm for the optimal scheduling of batch plants,” *International Journal of Applied Mathematics and Computer Sciences*, 13(4) University of Zielona Gora & Lubusky Scientific Society, Zielona Gora, Poland, pp. 527–536.

Hägele, M., Nilsson, K., Pires, J.N. and Bischoff, R. (2016) “Industrial robotics,” in *Springer handbook of robotics*. Springer, pp. 1385–1422.

Hawkins, K.P., Vo, N., Bansal, S. and Bobick, A.F. (2013) “Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration,” *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 499–506.

Hu, B. and Chen, J. (2017) “Optimal Task Allocation for Human–Machine Collaborative Manufacturing Systems,” *IEEE Robotics and Automation Letters*, 2(4), pp. 1933–1940.

Ji, Q., Lan, P. and Looney, C. (2006) “A probabilistic framework for modeling and real-time monitoring human fatigue,” *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans*, 36(5) IEEE, pp. 862–875.

Johannsmeier, L. and Haddadin, S. (2017) “A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes,” *IEEE Robotics and Automation Letters*, 2(1), pp. 41–48.

Knepper, R.A., Ahuja, D., Lalonde, G. and Rus, D. (2014) “Distributed assembly with and/or graphs,” *Workshop on AI Robotics at the Int. Conf. on Intelligent Robots and Systems (IROS)*.

Lamon, E., de Franco, A., Peternel, L. and Ajoudani, A. (2019) “A Capability-Aware Role Allocation Approach to Industrial Assembly Tasks,” *IEEE Robotics and Automation Letters*, 4(4) IEEE, pp. 3378–3385.

Lasota, P.A., Rossano, G.F. and Shah, J.A. (2014) “Toward safe close-proximity human-robot interaction with standard industrial robots,” *2014 IEEE*

International Conference on Automation Science and Engineering (CASE). IEEE, pp. 339–344.

Lee, D.Y. and DiCesare, F. (1994) "Scheduling flexible manufacturing systems using Petri nets and heuristic search," *IEEE Transactions on robotics and automation*, 10(2) IEEE, pp. 123–132.

Li, K., Liu, Q., Xu, W., Liu, J., Zhou, Z. and Feng, H. (2019) "Sequence Planning Considering Human Fatigue for Human-Robot Collaboration in Disassembly," *Procedia CIRP*, 83, pp. 95–104.

Liang, J. (2016) *Berkeley AutoLab yumipy Documentation*. Available at: https://berkeleyautomation.github.io/yumipy/ (Accessed: April 3, 2020).

Maiolino, P., Woolley, R., Branson, D., Benardos, P., Popov, A. and Ratchev, S. (2017) "Flexible robot sealant dispensing cell using RGB-D sensor and off-line programming," *Robotics and Computer-Integrated Manufacturing*, 48, pp. 188–195.

el Makrini, I., Merckaert, K., Lefeber, D. and Vanderborght, B. (2017) "Design of a collaborative architecture for human-robot assembly tasks," *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, pp. 1624–1629.

de Mello, L.S.H. and Sanderson, A.C. (1986) *AND/OR Graph Representation of Assembly Plans.*

Nau, D.S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J.W., Wu, D. and Yaman, F. (2003) "SHOP2: An HTN planning system," *Journal of artificial intelligence research*, 20, pp. 379–404.

Nikolakis, N., Sipsas, K., Tsarouchi, P. and Makris, S. (2018) "On a shared human-robot task scheduling and online re-scheduling," *Procedia CIRP*, 78, pp. 237–242.

Potvin, J.R. and Bent, L.R. (1997) "A validation of techniques using surface EMG signals from dynamic contractions to quantify muscle fatigue during repetitive tasks," *Journal of Electromyography and Kinesiology*, 7(2) Elsevier, pp. 131–139.

Ranz, F., Hummel, V. and Sihn, W. (2017) "Capability-based task allocation in human-robot collaboration," *Procedia Manufacturing*, 9 Elsevier, pp. 182–189.

Rashedi, E., Nezamabadi-Pour, H. and Saryazdi, S. (2009) "GSA: a gravitational search algorithm," *Information Sciences*, 179(13), pp. 2232–2248.

Riedelbauch, D. and Henrich, D. (2017) "Coordinating flexible human-robot teams by local world state observation," *Robot and Human Interactive*

Communication (RO-MAN), 2017 26th IEEE International Symposium on. IEEE, pp. 1000–1005.

Riedelbauch, D. and Henrich, D. (2019) "Exploiting a Human-Aware World Model for Dynamic Task Allocation in Flexible Human-Robot Teams," *2019 International Conference on Robotics and Automation (ICRA)*., pp. 6511–6517.

Rosebrock, A. (2014) *4 Point OpenCV getPerspective Transform Example*., *pyimagesearch* Available at: https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/ (Accessed: November 26, 2019).

Rosell, J. (2004) "Assembly and task planning using Petri nets: a survey," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 218(8), pp. 987–994.

Ruud de Jong (2017) *Calculate the area of intersection of two rotated rectangles in python*., *stackoverflow* Available at: https://stackoverflow.com/questions/44797713/calculate-the-area-of-intersection-of-two-rotated-rectangles-in-python/45268241#45268241 (Accessed: September 4, 2019).

Salunkhe, O., Stensöta, O., Åkerman, M., Berglund, Å.F. and Alveflo, P.-A. (2019) "Assembly 4.0: Wheel Hub Nut Assembly Using a Cobot," *IFAC-PapersOnLine*, 52(13) Elsevier, pp. 1632–1637.

Schröter, D., Jaschewski, P., Kuhrke, B. and Verl, A. (2016) "Methodology to identify applications for collaborative robots in powertrain assembly," *Procedia CIRP*, 55 Elsevier, pp. 12–17.

de Silva, L., Lallement, R. and Alami, R. (2015) "The HATP hierarchical planner: Formalisation and an initial study of its usability and practicality," *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, pp. 6465–6472.

Smith, T., Benardos, P. and Branson, D. (2020) "Assessing worker performance using dynamic cost functions in human robot collaborative tasks," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 234(1) SAGE Publications Sage UK: London, England, pp. 289–301.

Suzuki, T., Kanehara, T., Inaba, A. and Okuma, S. (1993) "On algebraic and graph structural properties of assembly Petri net," *[1993] Proceedings IEEE International Conference on Robotics and Automation*., pp. 507–514.

del Valle, C. and Camacho, E.F. (1996) "Automatic assembly task assignment for a multirobot environment," *Control engineering practice*, 4(7) Elsevier, pp. 915–921.

Wang, W., Li, R., Diekel, Z.M. and Jia, Y. (2018) “Robot action planning by online optimization in human–robot collaborative tasks,” *International Journal of Intelligent Robotics and Applications*, 2(2), pp. 161–179.

Wilson, J.R. and Sharples, S. (2015) *Evaluation of human work*. CRC press.

Wolfe, J., Marthi, B. and Russell, S. (2010) “Combined task and motion planning for mobile manipulation,” *Twentieth International Conference on Automated Planning and Scheduling*.

Yanco, H.A. and Drury, J. (2004) “Classifying human-robot interaction: an updated taxonomy,” *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*., Vol.3, pp. 2841–2846.

Yanco, H.A. and Drury, J.L. (2002) “A taxonomy for human-robot interaction,” *Proceedings of the AAAI Fall Symposium on Human-Robot Interaction*., pp. 111–119.

el Zaatari, S., Marei, M., Li, W. and Usman, Z. (2019) “Cobot programming for collaborative industrial tasks: an overview,” *Robotics and Autonomous Systems*, 116 Elsevier, pp. 162–180.

Zha, X.F. and Lim, S.Y.E. (2000) “Assembly/disassembly task planning and simulation using expert Petri nets,” *International Journal of Production Research*, 38(15) Taylor & Francis, pp. 3639–3676.

Zhang, W. (1989) “Representation of assembly and automatic robot planning by Petri net,” *IEEE Transactions on Systems, Man, and Cybernetics*, 19(2) IEEE, pp. 418–422.

Appendices

Appendix A: Derivation of the Synthetic Fatigue Variable for the Completion Time Recovery Model

An expected completion time generated using Eq. (4.1) at task iteration *i*+*v* is given by

$$E_{i+v,j} = t_{w,j} + \tau'_j \ln(i+v). \tag{A.1.1}$$

Rearranging Eq. (A.1.1) gives

$$E_{i+v,j} = t_{w,j} + \tau'_j \ln\left(i\left(1+\frac{v}{i}\right)\right) \tag{A.1.2}$$

which using the rules of logarithms gives

$$E_{i+v,j} = t_{w,j} + \tau'_j \left(\ln(i) + \ln\left(1+\frac{v}{i}\right)\right) \tag{A.1.3}$$

that can be expanded out to

$$E_{i+v,j} = t_{w,j} + \tau'_j \ln(i) + \tau'_j \ln\left(1+\frac{v}{i}\right). \tag{A.1.4}$$

Given that an expected completion time generated using Eq. (4.1) at task iteration *i* is given by

$$E_{i,j} = t_{w,j} + \tau'_j \ln(i) \tag{A.1.5}$$

substituting Eq. (A.1.5) into Eq. (A.1.4) gives

$$E_{i+v,j} = E_{i,j} + \tau'_j \ln\left(1+\frac{v}{i}\right). \tag{A.1.6}$$

Rearranging Eq. (A.1.6), gives the definition of the synthetic fatigue variable as

$$\tau'_j = \frac{(E_{i+v,j} - E_{i,j})}{\ln\left(1+\frac{v}{i}\right)}. \tag{A.1.7}$$

for subtask *j*, given the expected completion times for the worker at task iterations *i* and *i*+*v*.