Is Evolutionary Computation Evolving Fast Enough?

Graham Kendall, *School of Computer Science, University of Nottingham, UK and University of Nottingham Malaysia Campus, MALAYSIA*

***Abstract*— Evolutionary Computation (EC) has been an active research area for over 60 years, yet its commercial/home uptake has not been as prolific as we might have expected. By way of comparison, technologies such as 3D printing, which was introduced about 35 years ago, has seen much wider uptake, to the extent that it is now available to home users and is routinely used in manufacturing. Other technologies, such as immersive reality and artificial intelligence have also seen commercial uptake and acceptance by the general public. In this paper we provide a brief history of EC, recognizing the significant contributions that have been made by its pioneers. We focus on two methodologies (Genetic Programming and Hyper-heuristics), which have been proposed as being suitable for automated software development, and question why they are not used more widely by those outside of the academic community. We suggest that different research strands need to be brought together into one framework before wider uptake is possible. We hope that this position paper will serve as a catalyst for automated software development that is used on a daily basis by both companies and home users.**

I. Introduction

EVOLUTIONARY Computation (EC) has been part of the research agenda for at least 60 years. In a typical EC algorithm, a population of potential solutions is created and they compete for survival. The weakest (less fit) members of the population are killed off, and the remaining members are retained and copies made, which are mutated. This new population is then evaluated with the expectation that the population's average fitness improves over time, along with the best performing individual solution.

It is debatable whether EC has had the impact in the commercial sector that other technologies have had, which have seen much more visible adoption. 3D printing is changing the way that manufacturing is done and is also moving into the home, to the extent that almost anybody can carry out 3D printing. Immersive reality is on the verge of changing society, in ways that are not totally clear yet. What is apparent is that applications such as Pokemon Go have sparked interest into the challenges and opportunities that immersive reality brings [1], [2], [3], [4], [5]. Ubiquitous computing is becoming more prevalent, enabling users to access computing resources in ways that were unimaginable even just a few years ago. Artificial Intelligence (AI) is becoming part of our daily lives, whether that is competing against the best human players in board games [6] or helping make self driving cars a reality [7], [8], [9]. EC has not had the same penetration as other technologies. A specific example we draw upon in this paper is large scale software development which is, arguably, where EC is most needed.

In this paper, we will look back at what EC promised and will suggest some challenges that, if addressed, might further advance EC and enable its wider adoption.

Writing a scientific paper that utilizes an evolutionary approach based on a real world problem is not the same as using an evolutionary approach to address a real world problem. This may seem a pedantic statement but a paper considering a problem that is drawn from the real world is not the same as addressing the actual problem faced by industry. Looking at a sample of EC papers, which are labeled as "real world applications" (see Related Work, Section II), often shows that the problem being tackled is a problem that would be recognized as a real world problem. However, the algorithm is often tested on benchmark datasets and/or uses a simplified model of the problem. It is our view that if a problem is presented as a real world problem, there should be an underlying model that addresses a real problem faced by the user community, rather than a simplified model that is an abstraction of the problem that users actually face.

We do recognize the importance that benchmark datasets play in the investigation, and development of, algorithmic approaches. Indeed, many important breakthroughs have been reported by investigating these real world abstractions. We also note that studying these simplified problems enables easier analysis of the results. We are also aware that this type of research (using a simplified problem) is why games are often used, as they have fixed rules, the rules are unambiguous and there is a winner and a loser. It is no coincidence that Chess has been called the drosophila of artificial intelligence [10]. We are also conscious that abstracting a problem so that the focus is on the methodology is good scientific research and that fully modeling a specific problem may not be of benefit to the wider academic community.

So, we are not critical of using abstractions of the real world but it is not conducive to promoting EC to the commercial sector, who require solutions to problems which go beyond these benchmarks, and which address the needs of their specific business.

The fact that the EC research community does not tend to tackle real, real world problems is partly (largely?) due to the industrial/university communities not working together.

Digital Object Identifier: 10.1109/???.????.???????
Date of publication: dd month year

Corresponding Author: Graham Kendall (Email: Graham.Kendall@nottingham.ac.uk)

This is not a criticism of the companies, or the universities. Universities and companies often have different objectives (e.g. carrying out research vs. making a profit) and they work on different time scales (e.g. long term research projects vs developing new products to maintain a competitive edge). Another contributory factor may be that other methodologies may be better accepted by the commercial sector as they are easier to understand, implement and support. In this respect, the use of methodologies such as EC can be seen to be similar to a reluctance to utilize Artificial Neural Networks. The decisions they reach are not easy to understand so that the commercial sector is often unwilling to adopt them, preferring methodologies where the decisions can be more easily explained.

This paper is structured as follows. In the next section we consider related work, focusing on those papers that have reported using EC on real world problems. In Section III we ask if the potential of EC has been achieved? We note the significant achievements of the EC pioneers and ask why their seminal work has not translated into more uptake outside of the scientific community. Section IV looks at Genetic Programming (GP), bearing in mind it was probably this EC methodology that had (has?) the most promise to be used in the commercial sector. In Section V we consider a more recent methodology (Hyper-heuristics) which also has the potential to be used by industry. In Section VI we look specifically at Large Scale Software Development, highlighting some of its high profile failures and asking if/how EC can help in this area. In Section VII we present some suggested research directions, before concluding in Section VIII.

II. Related Work

The topic of deploying evolutionary algorithms in the real world has been studied before [11]. Within the context of this paper, [11] provides a number of inhibitors to using EC based algorithms in the real world. These include:

- The features of real world problems
- The lack of faith in the underlying model that represents the problem meaning that companies have little confidence in the solutions that are produced
- The fact that EC algorithms are not integrated within an overarching framework to assist with areas such as parameter settings
- The lack of the required skills of the developers
- Resistance to change

It is difficult to track down examples in the scientific literature where EC has been deployed in a company and is used as a matter of course in their routine activities. Of course, there will be examples that the scientific community is not aware of due to in-house research activity, commercial sensitivities or the time pressures within a company to write and present the contribution to the scientific community, but we do not believe that the use of EC has had large adoption within commercial companies. There are many examples (e.g. [12], [13], [14]) where real data has been used and the results are encouraging, but the algorithm has not been used by a company to support its on-going business activities. It is often the case that a company provides data, which is utilized to study the problem, it is then reported in the scientific literature but the algorithmic methodology is not used by the company.

A Genetic Algorithm (GA) [15] was reported in [16], presenting a two-phase algorithm for the "bid-line generation problem" (the problem of scheduling airline crew) for Delta Air Lines. As with many staff scheduling problems, there are many industry and legal factors to take into account [17], [18] so any systems that are developed for a given company are often bespoke. The first phase of their algorithm generates as many high-quality lines as possible. The second phase, where the GA is run, completes the assignments. The schedules that are produced were shown to be of comparable quality to those that were generated using a semi-automatic process that the airline had previously used. It is interesting to note that the four authors listed their affiliations as Delta Technology or Delta Airlines, indicating that the paper was written without a university collaborator. This, we feel, is important as it demonstrates that the industrial sector is deploying EC algorithms. However, many companies will not report these successes in the scientific literature for a number of reasons including lack of time, no pressure to publish and commercial confidentiality. These factors are likely to misrepresent the real scale of industrial take up of EC methodologies in industry, and the lack of reporting in the scientific literature could slow down progress.

Sundararajan et al. [19] considered the cross selling of loans in the banking sector, specifically the GEMB bank in Poland. They used a GA, within an overall framework which draws on different methodologies, which focussed on a predictive model for response, risk and profit. The GA that was developed was a standard GA with a few enhancements that included elitism and splitting the data into training and validation sets and using solutions from one set to inject into the other set if it finds that it performs well. Similar to [16], this paper also had no authors with a university affiliation.

A GA was also utilized in [20]. The two authors were from Intel, with no university affiliation listed. They presented a model for the Product Line Design and Scheduling Problem. The outer layer of their model was a GA. This handled the resource constraints, scheduling, and financial optimization. An inner layer utilized mathematical programming to optimize product composition. Their new approach replaced a spreadsheet solution which could take days, or even weeks, to carry out what-if analysis.

The gerrymandering problem (the process of manipulating electoral boundaries to gain a political advantage) was addressed in [21]. This was a joint paper between university colleagues and a representative from a government department (Philadelphia Water Department's Office of Watersheds). This paper was written as the result of a competition call. The authors won one of the competition categories which gave them the opportunity to present to the city council. The authors classify their algorithm as a form of Evolutionary Programming, rather than as a GA as they did not use a recombination operator.

A recent paper [22], a collaboration between two universities and Ernst & Young, considered the transportation

and scheduling issues for the 2014 Special Olympics USA Games. The problem considered 3,300 athletes with intellectual disabilities, 1,000 coaches and over 70,000 spectators. The athletes competed in 16 sports, across 10 locations, spread over a 30-mile radius. The authors developed a GA to address the problem as exact methodologies were too computationally expensive. The resulting schedules were used during the games.

Another routing problem was addressed in [23], in a paper that did not include a company representative as an author. The paper presents a case study based on a humanitarian scenario, a local branch of the Meals on Wheels Association of America, which provides food to individuals who are in need. The approach adopted interfaces a spreadsheet with a GA and is being used by the Metro Meals on Wheels Treasure Valley. It is noted that the tool could be used anywhere that has access to Google Maps or MapQuest.

Ogris et al. [24] studied a primary school timetabling problem [25] from Slovenia. The paper was co-authored by university researchers and industrial collaborators. Their evolutionary algorithm (there was no crossover operator) comprised three objective functions, which were changed probabilistically. The system was used in three Slovenian primary schools but could easily be adapted to other schools and universities.

Simulated annealing [26] and Tabu search [27] are not classified as evolutionary methodologies, rather they are metaheuristics [28]. However, they have been used in industrial applications so we thought it was worth briefly mentioning them here. We also note that the leading EC journal (IEEE Transactions on Evolutionary Computation) has previously reported work that includes these methodologies [29], [30], albeit hybridized with an evolutionary algorithm. Simulated annealing and tabu search has been reported as being deployed in industry, including Oil Field Drilling [31], Sports [32], [33], [34], [35], Vehicle Routing [36], Underground Mine Layouts [37] and Personnel Scheduling [38].

The papers that we discuss above might suggest that there has been a lot of commercial applications of EC but considering that the field has been active for over 60 years, the number of reported applications of EC methodologies is somewhat small. No doubt, there are other papers that we have not included and there will be successes that are not reported in the scientific literature but we still argue that adoption of EC by the commercial sector is not as prevalent as some other technologies.

This might be about to change with recent interest in Deep Learning and the success of projects such as AlphaGo [6], which was able to win against the best Go player in the world, a feat that most people predicted would take another ten years. However, this is just one methodological example and, whilst Deep Learning Neural Networks have a bright future, it still does not answer the question as to why more EC methodologies have not had wider uptake.

III. Has the potential been realized?

EC has been an active research area since the 1950's [39], [40]. Many eminent scientists have been recognized as being pioneers in this field, demonstrating the strength in depth of this area. Table I shows the IEEE Computational Intelligence Society Pioneers, along with a small sample of their contributions. It is beyond question that these pioneers, along with the wider community, have made significant advances in EC.

When these pioneers were carrying out their early work, it was in their minds that it would be adopted by the wider community. For example, Box [41] says "Its basic philosophy is that it is nearly always inefficient to run an industrial process to produce product alone. A process should be run so as to generate product plus information on how to improve the product." In 1996, Schwefel said "...the past decade has witnessed an exponential increase in diverse applications, from design synthesis, planning and control processes, to various other adaptation and optimization tasks."

It is, perhaps, surprising that we have not seen more examples reported in the scientific literature of EC being deployed in commercial systems. Although the related work section provides some examples, and no doubt some are missing, but given that the field has been active for at least 60 years we might expect to see more examples being reported?

In comparison, 3D printing [91], [92] has seen a significantly faster uptake. The first patent was issued to Charles Hull in 1986, which can be traced back to his original invention from 1983. Since then the technology has seen rapid uptake, to the point where it is now possible to buy a 3D printer for home use. It is likely that we are only just seeing the start of the additive manufacturing technology and it is likely that many replacement parts, rather than being bought at a shop, or online, can be downloaded and printed at home. By comparison, the software development industry is not able to offer the home user a way to develop, or evolve, software unless they are already skilled programmers or willing to invest a significant amount of time learning a programming language.

Technologies such as GP and Hyper-heuristics (both discussed below), despite delivering excellent research advances, have not really made the transition from the research environment to a position where the benefits can be experienced by an average home user. In the next two sections, we focus on these two methodologies, though similar analysis could be made of the many other EC variants that have been researched over the years.

IV. Genetic Programming

Many of the papers that were discussed in Section II utilized GAs, yet GP is, arguably, the EC methodology that is most associated with automated software development.

Introduced by Koza [93], [94], [95], GP seeks to evolve computer programs and/or evolve functions. Does it matter which it does; evolve programs or functions?

In [96] the authors say (Section 1.1) "In genetic programming we evolve a population of computer programs." In one of the seminal GP papers [93], it states "Automatic programming requires developing a computer program that can produce a desired output for a given set of inputs", which is more akin to suggesting that GP evolves functions, rather than a program. We can debate whether a function (a relationship between a

TABLE I
IEEE COMPUTATIONAL INTELLIGENCE SOCIETY EVOLUTIONARY COMPUTATION PIONEERS

Year	Pioneer	References
2016	Marco Dorigo	[42], [43], [44]
2015	Thomas Bäck	[45], [46], [47]
2014	George Burgin	[48], [49]
2013	Xin Yao	[50], [51], [52], [53]
2012	Russell C. Eberhart, James Kennedy, and J. David Schaffer	[54], [55], [56], [57], [58], [59], [60]
2011	Larry J. Eshelman	[55], [56], [57]
2010	David E. Goldberg and John Grefenstette	[61], [62], [63]
2008	David B. Fogel	[64], [65], [66], [67], [47], [68]
2005	Kenneth De Jong	[69], [70], [71], [72]
2004	Richard Friedberg	[73], [74]
2003	John H. Holland	[75], [76], [61], [62]
2002	Ingo Rechenberg and Hans-Paul Schwefel	[77], [78], [45], [79], [80]
2001	Michael Conrad	[81], [82], [83]
2000	George Box	[41], [84]
1999	Alex S. Fraser	[40], [85], [86], [87]
1998	Lawrence J. Fogel	[39], [88], [89], [48], [49], [90]

set of inputs and a permissible set of outputs) and a program (a sequence of coded instructions to automate a task on a computer) are the same thing but to the general public if GP is sold as evolving computer programs they will assume that this means that a complete program will be evolved, and not just a function (a mathematical function or a function for a given programming language), which is usually the case. We hasten to add that no criticism is implied, or meant, of the GP pioneers, or other researchers. The terminology has evolved over time and the expressions used in the scientific literature are the ones that are most applicable, or preferred, by the authors of a given paper. We note, as in many areas of EC — and even beyond, such as the heuristic community — that there are no widely accepted terms and definitions in much of the terminology that is used.

However, to the general public saying "evolve computer programs" may indicate that GP is much more general than the state of the art would suggest. There have been advances in moving towards more general environments. The 2016 Human Competitive Awards, the so called "Humies"[1], winner [97] says "Automated transplantation would open many exciting avenues for software development: suppose we could auto-transplant code from one system into another, entirely unrelated, system. This paper introduces a theory, an algorithm, and a tool that achieves this." This is certainly a significant contribution to automated program development but there is still a lot of work to do, as acknowledged by the authors, "While we do not claim automated transplantation is now a solved problem, our results are encouraging."

Since 2004, the GP community has been able to compete in the Humies. This annual competition invites entries that report human-competitive results by any form of genetic or evolutionary computation. The entries must satisfy one of the following eight criteria (taken from[1]):

1) The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.
2) The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.
3) The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.
4) The result is publishable in its own right as a new scientific result independent of the fact that the result was mechanically created.
5) The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.
6) The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.
7) The result solves a problem of indisputable difficulty in its field.
8) The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

The Humies have certainly demonstrated the versatility of GP (see Table II), along with other EC approaches. However, looking at the papers, which support the entries, shows that GP still requires tailoring for the problem at hand. It might also be argued that some of the problems are not challenging, with respect to the domains that they address and the fact that they do not suggest that they have a more generic applicability.

There are GP frameworks available, but they still require the knowledge and experience of the researcher to utilize that framework and then tailor it for the problem under consideration. Unquestionably, GP has succeeded, and continues to do so and the scientific literature has a significant body of peer reviewed work on this topic. However, it has yet to get to the position where it can be used by a non–expert user, sitting at home, who wants to evolve software for a problem they have.

[1]http://www.human-competitive.org/awards, last accessed 04 Feb 2018

TABLE II
HUMIES GOLD MEDAL WINNERS (IN SOME YEARS THE GOLD MEDAL WAS SHARED, INDICATED BY "=")

Year	Entry	References
2017	"Explaining quantum correlations through evolution of causal models"	[98]
2016	"Automated Software Transplantation"	[97]
2015	"Evolutionary Approach to Approximate Digital Circuits Design"	[99]
2014	"Genetic Algorithms for Evolving Computer Chess Programs"	[100]
2013=	"Evolutionary Design of FreeCell Solvers"	[101]
2013=	"Search for a grand tour of the Jupiter Galilean moons"	[102]
2012	"Go without KO on Hexagonal Grids" and "Yvalath: Evolutionary Game Design"	[103]
2011	"GA-FreeCell: Evolving Solvers for the Game of FreeCell"	[104]
2010	"Evolutionary design of the energy function for protein structure prediction" and "GP challenge: evolving the energy function for protein structure prediction" and "Automated design of energy functions for protein structure prediction by means of genetic programming and improved structure similarity assessment"	[105], [106], [107]
2009	"Automatically finding patches using genetic programming" and "A Genetic Programming Approach to Automated Software Repair"	[108], [109]
2008	"Genetic Programming for Finite Algebras"	[110]
2007	"Evolutionary Design of Single-Mode Microstructured Polymer Optical Fibres using an Artificial Embryogeny Representation"	[111]
2006	"Catalogue of Variable Frequency and Single-Resistance-Controlled Oscillators Employing A Single Differential Difference Complementary Current Conveyor" and "Novel Canonic Current Mode DDCC Based SRCO Synthesized Using a Genetic Algorithm" and "Evolving Sinusoidal Oscillators Using Genetic Algorithms"	[112], [113], [114]
2005=	"Two-dimensional photonic crystals designed by evolutionary algorithms"	[115]
2005=	"Learning from Learning Algorithms: Applications to attosecond dynamics of high-harmonic generation" and "Shaped-pulse optimization of coherent soft-x-rays"	[116], [117]
2004=	"An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission"	[118]
2004=	"Automatic Quantum Computer Programming: A Genetic Programming Approach"	[119]

V. HYPER-HEURISTICS

A hyper-heuristic has the aim of raising the level of generality of search/optimization algorithms, recognizing that no one search algorithm exists that is superior across all search/optimization problems [120]. Instead of searching the solution space directly, the most relevant heuristic to apply at any decision point is identified, which is applied to the solution space. It is hoped that a hyper-heuristic search algorithm can be applied to a wide range of problems, simply by changing the heuristics and utilizing the same heuristic search algorithm. Following these so called "Heuristic Selection Algorithms", later research investigated whether the heuristics themselves could be evolved [121], [122] thus saving the need to implement heuristics when new problems are tackled.

The first mention of the term "hyper-heuristic" in the scientific literature was in [123] (the term was also used in [124], but in a different context), although even earlier work could also be regarded as being a hyper-heuristic (e.g. [125], [126]), although the term was not used. A survey of hyper-heuristics is available in [127].

A 2000 research proposal (the author of this paper was one of the authors) said: "We will try to demonstrate how quick- and cheap-to-implement knowledge-poor heuristics can be used within a hyper-heuristic framework to provide a methodology suited to fast and cheap development of industrial and commercial systems. This will lead to a prototype hyper-heuristic 'toolbox' for the user community."

The authors of the proposal recognized that to provide a methodology suited to fast and cheap development of industrial and commercial systems was a challenging goal, and it was recognized that it would not be completed in the lifetime of the research award but, nonetheless, it was a long term vision for hyper-heuristics.

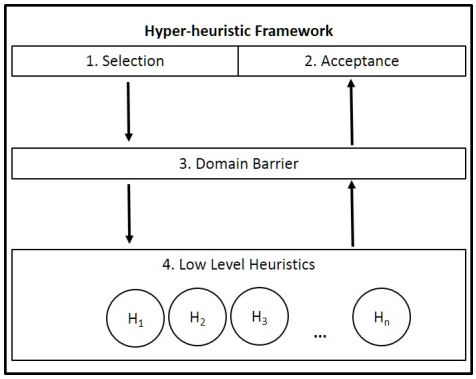

Fig. 1. Hyper-Heuristic Framework.

A generic hyper-heuristic framework is shown in Fig. 1. The initial research in hyper-heuristics focused on methodologies where several low-level heuristics were provided (no. 4 in the figure) and a high-level selector (no. 1) chooses which of the low level heuristics to apply at any given decision point. This was the so called "Heuristics to Choose Heuristics." Note should be taken of the domain barrier (no. 3). The high-level selector has no knowledge of the domain. Rather, it only knows how many heuristics there are and receives non-domain feedback, such as change in evaluation function, computation time etc. This enables the high level selector to operate on different domains, by replacing the low level heuristics by those that are able to address the new problem at hand.

Having to develop and replace a set of low level heuristics led to the obvious research question; can we evolve the low

level heuristics so that we do not have to implement them when we want to change domains? A further question is, should a solution from one of the low level heuristics being accepted as the incumbent solution, and what form should that acceptance criteria take (e.g. always accept, improving only, sometimes accept worse solutions etc.) and can this acceptance criteria be evolved?

These latter questions are of more interest to the focus of this paper, as the approaches tend to be more EC based, and these research directions have been investigated in recent papers (e.g. [128], [129], [130]).

Hyper-heuristics have been an active research area for at least 20 years, and arguably back to the 1960's, yet there is still no off-the-shelf hyper-heuristic product that enables the commercial sector to benefit from this technology, let alone home users being able to access this methodology in the same way that they can now access 3D printing and immersive reality.

VI. Large Scale Software Development

As noted in Section IV, GP has had many successes and hyper-heuristic research (Section V) has made significant progress in the last 20 years. Both technologies still have some way to go before being able to be offered to the business/home user in an easy to use form.

The scientific community recognizes that GP evolves functions, and saying that it evolves programs, could be viewed in a different way by the non-GP community, which means that their expectations are not met when they start using GP as a tool to integrate with their own systems.

Hyper-heuristic research has tended to focus on the main elements of the framework (see Fig. 1). There has been some work in trying to unify the various elements, but nothing is readily available at the moment that can be used off-the-shelf.

There are tools available, such as TSPLIB[2], MATLAB[3] and CPLEX[4] but these are either expensive, more suited to expert users and not necessarily EC related.

We know that large scale software development is difficult. Rosenberg [131] tells the story of Mitch Kapor who developed Lotus 1-2-3 and the popular personal information manager, Agenda. Kapor decided to develop a more up to date, extensible, fully functioning and featured personal information manager. What started as a grand vision became a tale of managing a large software development team with all the issues and problems that this brings. The resultant product, Chandler, is freely available but it never had the impact that was hoped for. The book [131] provides a stark reference to the difficulties of large scale software development, even by people who have developed highly successful products before.

Brooks [132], in his famous work – The Mythical Man–Month – noted that software development is difficult and when large software development projects do run into problems, adding additional manpower cannot save it. Indeed, it will make it even later.

There are many examples of software development projects failing. A small sample (there are numerous) are highlighted here:

1) "The U.S. Air Force has decided to scrap a major ERP (enterprise resource planning) software project after spending US$1 billion, concluding that finishing it would cost far too much more money for too little gain."[5]
2) "In 2003, Levi Strauss, was a global corporation, with operations in more than 110 countries but with an IT system that was an antiquated, 'Balkanised' mix of incompatible country-specific systems. So its bosses decided to migrate to a single SAP system and hired a team of fancy consultants (from Deloitte) to lead the effort. 'The risks seemed small,' wrote the researchers. 'The proposed budget was less than $5m.' But very quickly things fell apart. One major customer, Walmart, required that the system interface with its supply chain management system, creating additional work. During the switchover to the new system, Levi Strauss was unable to fulfil orders and had to close its three US distribution centres for a week. In 2008, the company took a $192.5m charge against earnings to compensate for the botched project and fired its chief information officer."[6]
3) "We examined 1,471 projects, comparing their budgets and estimated performance benefits with the actual costs and results. They ran the gamut from enterprise resource planning to management information and customer relationship management systems. Most, like the Levi Strauss project, incurred high expenses - the average cost was $167 million, the largest $33 billion – and many were expected to take several years. Our sample drew heavily on public agencies (92%) and U.S.-based projects (83%), but we found little difference between them and projects at the government agencies, private companies, and European organizations that made up the rest of our sample."[7]

There appears to be a need for more support for large scale software development projects. There are enough personnel working as software developers (see Table III[8]) that any automation should be welcomed by the industry. Perhaps not by those whose jobs are at risk, but certainly by those who employ the developers. Of course, this is no different to many other industries, where jobs have been replaced by automation, but it does seem ironic that those responsible for automating so many jobs are now at risk themselves.

Even if we were able to get technologies such as GP and hyper-heuristics to the stage where they could be used by experienced software developers, it is not clear how these technologies could be packaged to make them readily available to business/home users, who are not experienced developers.

[2]https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/, last accessed 04 Feb 2018

[3]https://www.mathworks.com/, last accessed 04 Feb 2018

[4]https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer, last accessed 04 Feb 2018

[5]https://www.cio.com/article/2390341/, last accessed 04 Feb 2018

[6]https://www.theguardian.com/technology/2013/apr/21/fred-brooks-complex-software-projects, last accessed 04 Feb 2018

[7]https://hbr.org/2011/09/why-your-it-project-may-be-riskier-than-you-think, last accessed 04 Feb 2018

[8]https://www.infoq.com/news/2014/01/IDC-software-developers, last accessed 04 Feb 2018

TABLE III
SIZE OF SOFTWARE DEVELOPER COMMUNITY, FROM IDC STUDY

Role		Estimated # for 2014
ICT-skilled Workers	Professional software developers	11,005,000
	ICT Operations and management skilled workers	18,008,900
	Total	29,013,900
Software Developers	Professional software developers	11,005,000
	Hobbyist software developers	7,534,500
	Total	18,539,500

It is unrealistic, at least in the foreseeable future, to expect an evolutionary process to evolve a complete software product and perhaps this will never be an aim, or an expectation. Perhaps a more immediate aim would be to enable software developers to specify the requirements and interface as part of the software development life cycle and let an evolutionary process deliver required functionality, with some guarantees that it is fit for the purpose.

Indeed, this is similar to evolutionary art [133], which has a long and proven history [134]. In this paradigm the human is often part of the fitness evaluation and they judge the quality of the art that the evolutionary process produces. It could be envisaged that humans judge the quality of an evolved program by being part of the fitness evaluation. In this way, the human developer would not simply be a coder but would be tasked with guiding the evolutionary process via their feedback as to the effectiveness of each member of the population and would be helping to decide which programs are worthy of surviving to the next generation.

This could be seen as being part of an agile approach to software development. That is, software developers start developing the software system by providing some functionality and gradually adding to it. If they come across some part of the system that is particularly difficult to develop they could call upon an EC based approach to evolve the required functionality, perhaps while they work on other parts of the system. Once the required functionality has been evolved, it is simply plugged into the system without the software developer having to do anything else. This functionality could even continue to evolve, should that be required, even when the system is deployed in a live environment.

It is likely that we would also have to draw on "Search-based Software Engineering" (i.e. the utilization of search methodologies such as GAs, simulated annealing and tabu search to address software engineering problems) [135], [136], [137]. If we are able to develop a user friendly framework that incorporates EC, search based software engineering; along with guarantees of what is delivered this would be a powerful product which would benefit the wider world, outside of the scientific community.

VII. SUGGESTED RESEARCH DIRECTIONS

Despite the large number of references in this paper, a more extensive survey of where EC has been used in the real world would certainly of benefit, if nothing else to serve as a baseline for future researchers. It would be useful to carry out a survey/analysis considering which methodologies from the scientific literature are utilized by the industrial community and to understand the reasons why some methodologies are adopted, whilst others are not. It would also be useful to survey the existing scientific literature to establish when authors say they are addressing a real world problem, is this really the case or are they modelling a simplified version of a problem, utilizing a benchmark dataset or addressing a problem that would not be recognized by the industrial community?

Most of the examples given utilized GAs. This is a little surprising as there are many other methodologies available [138], although GAs were one of the earliest and most popular EC methodologies. There might be some scope to look at how industry could benefit from other methodologies, as well as reporting non-GA examples that have been successfully deployed in industry. A book, or a series of articles, aimed at the commercial community might be useful so that there could be more take up.

The scientific community may benefit from a more complete survey where EC has been used in applications outside of the research arena. This might provide insights into the most useful methodologies, what domains are taking up the use of EC and the benefits that have arisen from using EC in a commercial environment.

Frameworks, that could be used out of the box, would be a valuable addition to the tools available to the commercial sector. It is recognized that some of these tools do exist but it is a steep learning curve, and sometimes expensive, for inexperienced users to start using them.

It would certainly be useful to investigate how various methodologies, such as EC, hyper-heuristics and search based software engineering could be integrated into a single framework.

If there was an integrated framework that enabled EC to be made easily available to the industrial/home user, it begs the question which EC methodology would be most suitable to use for a given problem provided to the framework? This is certainly worthy of further research. That is, provided with a problem should the framework use GA, GP; or one of the many other EC methodologies that are available, or even hybridizations of two, or more, of them?

VIII. CONCLUSION

The related work section of this paper has highlighted a number of projects where EC has been used, and is being used, in applications that have been deployed in the real world. It is noticeable that there are relatively few papers which report deployment of EC into a live industrial environment. It is also noticeable that many of these papers are from R&D departments within the companies involved.

We are certainly a long way from where an interested home user can access EC in the same way that access to 3D printing and immersive reality have become possible in the past few years.

EC has made significant research progress in the past 60 years but an integrated framework is lacking where all of this functionality can be easily accessed. The development of a framework would be welcome but there is research activity that needs to take place to support this framework so that the underlying complexity remains largely hidden from the end user.

ACKNOWLEDGEMENT

This paper is based on a plenary talk that the author gave at the 2016 Congress on Evolutionary Computation, 24-29 July 2016, Vancouver. The author would like to acknowledge the support of IEEE and the Computational Intelligence Society, as without the opportunity to present the plenary talk it would not have been possible to write this paper.

REFERENCES

[1] N. R. Murch, "Game on for Pokemon Go: Placement of Pokemon characters may breach confidentiality," *BMJ – British Medical Journal*, vol. 354, 2016.

[2] T. Althoff, R. W. White, and E. Horvitz, "Influence of Pokemon Go on physical activity: Study and implications," *Journal of Medical Internet Research*, vol. 18, no. 12, pp. 82–95, 2016.

[3] M. Tateno, N. Skokauskas, T. A. Kato, A. R. Teo, and A. P. S. Guerrero, "New game software (Pokemon Go) may help youth with severe social withdrawal, hikikomori," *Phychiatry Research*, vol. 246, pp. 848–849, 2016.

[4] M. Serino, K. Cordrey, L. McLaughlin, and R. L. Milanaik, "Pokemon Go and augmented virtual reality games: A cautionary commentary for parents and pediatricians," *Current Opinion in Pediatrics*, vol. 28, no. 5, pp. 673–677, 2016.

[5] J. W. Ayers, E. C. Leas, M. Dredze, J.-P. Allem, J. G. Grabowski, and L. Hill, "Pokemon GO - A new distraction for drivers and pedestrians," *Jama Internal Medicine*, vol. 176, no. 12, pp. 1865–1866, 2016.

[6] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[7] S. Bringsjord and A. Sen, "On creative self-driving cars: Hire the computational logicians, fast," *Applied Artificial Intelligence*, vol. 30, no. 8, pp. 758–786, 2016.

[8] N. J. Goodall, "Can you program ethics into a self-driving car?" *IEEE Spectrum*, vol. 53, no. 6, pp. 28–58, 2016.

[9] S. E. Shladover, "The truth about "self-driving" cars," *Scientific American*, vol. 314, no. 6, pp. 53–57, JUN 2016.

[10] N. Ensmenger, "Is chess the drosophila of artificial intelligence? A social history of an algorithm," *Social Studies of Science*, vol. 42, no. 1, pp. 5–30, 2012.

[11] V. Oduguwa, A. Tiwari, and R. Roy, "Evolutionary computing in manufacturing industry: An overview of recent applications," *Applied Soft Computing*, vol. 5, no. 3, pp. 281–299, 2005.

[12] N. M. Mohmad Kahar and G. Kendall, "A great deluge algorithm for a real-world examination timetabling problem," *Journal of the Operational Research Society*, vol. 66, no. 1, pp. 116–133, 2015.

[13] N. G. Beligiannis, C. Moschopoulos, and D. S. Likothanassis, "A genetic algorithm approach to school timetabling," *Journal of the Operational Research Society*, vol. 60, no. 1, pp. 23–42, 2009.

[14] G. T. Dias, P. J. de Sousa, and F. J. Cunha, "Genetic algorithms for the bus driver scheduling problem: A case study," *Journal of the Operational Research Society*, vol. 53, no. 3, pp. 324–335, 2002.

[15] K. Sastry, D. E. Goldberg, and G. Kendall, *Introductory Tutorials in Optimization and Decision Support Techniques*. Springer US, 2014, ch. Genetic Algorithms, pp. 93–117.

[16] I. Christou, A. Zakarian, J.-M. Liu, and H. Carter, "A two-phase genetic algorithm for large-scale bidline-generation problems at Delta Air Lines," *Interfaces*, vol. 29, no. 5, pp. 51–65, 1999.

[17] J. Van den Bergh, J. Belin, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: A literature review," *European Journal of Operational Research*, vol. 226, no. 3, pp. 367–385, 2013.

[18] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, vol. 153, no. 1, pp. 2–27, 2004.

[19] R. Sundararajan, T. Bhaskar, A. Sarkar, S. Dasaratha, D. Bal, J. K. Marasanapalle, B. Zmudzka, and K. Bak, "Marketing optimization in retail banking," *Interfaces*, vol. 41, no. 5, pp. 485–505, 2011.

[20] E. Rash and K. Kempf, "Product line design and scheduling at intel," *Interfaces*, vol. 42, no. 5, pp. 425–436, 2012.

[21] R. Gopalan, S. O. Kimbrough, F. H. Murphy, and N. Quintus, "The philadelphia districting contest: Designing territories for city council based upon the 2010 census," *Interfaces*, vol. 43, no. 5, pp. 477–489, 2013.

[22] A. Johnson, Y. Zhao, and X. Xu, "Transportation planning and scheduling for the 2014 special olympics usa games," *Interfaces*, vol. 46, no. 3, pp. 218–230, 2016.

[23] A. S. Manikas, J. R. Kroes, and T. F. Gattiker, "Metro meals on wheels treasure valley employs a low–cost routing tool to improve deliveries," *Interfaces*, vol. 46, no. 2, pp. 154–167, 2016.

[24] V. Ogris, T. Kristan, A. Škraba, M. Urh, and D. Kofjač, "iUrnik: Timetabling for Primary Educational Institutions in Slovenia," *Interfaces*, vol. 46, no. 3, pp. 231–244, 2016.

[25] N. Pillay, "A survey of school timetabling research," *Annals of Operations Research*, vol. 218, no. 1, pp. 261–293, 2014.

[26] E. Aarts, J. Korst, and W. Michiels, *Introductory Tutorials in Optimization and Decision Support Techniques*. Springer US, 2014, ch. Simulated Annealing, pp. 265–285.

[27] M. Gendreau and J.-Y. Potvin, *Introductory Tutorials in Optimization and Decision Support Techniques*. Boston, MA: Springer US, 2014, ch. Tabu Search, pp. 243–263.

[28] E. Burke and G. Kendall, Eds., *Introductory Tutorials in Optimization and Decision Support Techniques*. Springer US, 2014.

[29] R. Bai, E. K. Burke, G. Kendall, J. Li, and B. McCollum, "A hybrid evolutionary approach to the nurse rostering problem," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 580–590, 2010.

[30] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269–283, 2008.

[31] K. Eagle, "Using simulated annealing to schedule oil field drilling rigs," *Interfaces*, vol. 26, no. 6, pp. 35–43, 1996.

[32] M. A. Trick, H. Yildiz, and T. Yunes, "Scheduling major league baseball umpires and the traveling umpire problem," *Interfaces*, vol. 42, no. 3, pp. 232–244, 2012.

[33] A. Farmer, J. S. Smith, and L. T. Miller, "Scheduling umpire crews for professional tennis tournaments," *Interfaces*, vol. 37, no. 2, pp. 187–196, 2007.

[34] J. R. Willis and J. B. Terrill, "Scheduling the australian state cricket season using simulated annealing," *Journal of the Operational Research Society*, vol. 45, no. 3, pp. 276–280, 1994.

[35] M. Wright, "Timetabling county cricket fixtures using a form of tabu search," *Journal of the Operational Research Society*, vol. 45, no. 7, pp. 758–770, 1994.

[36] M. J. Fry and J. W. Ohlmann, "Route design for delivery of voting machines in hamilton county, Ohio," *Interfaces*, vol. 39, no. 5, pp. 443–459, 2009.

[37] M. Brazil, P. Grossman, J. H. Rubinstein, and D. Thomas, "Improving underground mine access layouts using software tools," *Interfaces*, vol. 44, no. 2, pp. 195–203, 2014.

[38] K. W. Campbell, R. B. Durfee, and G. S. Hines, "Fedex generates bid lines using simulated annealing," *Interfaces*, vol. 27, no. 2, pp. 1–16, 1997.

[39] L. J. Fogel, "The human computer in flight control," *I.R.E. Transactions on Electronic Computers*, vol. EC-6, no. 3, pp. 197–202, 1957.

[40] A. S. Fraser, "Simulation of genetic systems by automatic digital computers. I. Introduction," *Australian Journal of Biological Sciences*, vol. 10, pp. 484–491, 1957.

[41] G. E. P. Box, "Evolutionary operation: A method for increasing industrial productivity," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 6, no. 2, pp. 81–101, 1957.

[42] M. Dorigo, M. Vittorio, and C. Alberto, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man and Cybernetics Part B : Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[43] M. Dorigo, G. Di Caro, and L. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[44] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behavior," *Nature*, vol. 406, pp. 39–42, 2000.

[45] T. Bäck and H. Schwefel, "Evolutionary computation: An overview," in *International Conference on Evolutionary Computation*, 1996, pp. 20–29.

[46] T. Bäck, "An overview of parameter control methods by self-adaptation in evolutionary algorithms," *Fundamenta informaticae*, vol. 35, pp. 51–66, 1998.

[47] T. Bäck, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. IOP Publishing and Oxford University Press, 1998.

[48] L. Fogel and G. Burgin, "Competitive goal-seeking through evolutionary programming," in *Final Report, Contract AF 19(628), Air Force Cambridge Research Laboratories*, 1969.

[49] G. Burgin and L. Fogel, "Air-to-air combat tactics synthesis and analysis program based on an adaptive maneuvering logic," *Journal of Cybernetics*, vol. 2, no. 4, pp. 60–68, 1972.

[50] X. Yao, "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, vol. 8, no. 4, pp. 539–567, 1993.

[51] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.

[52] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[53] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[54] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, 1985, pp. 93–100.

[55] J. Schaffer, R. Caruana, L. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proceedings of the third international conference on Genetic algorithms*, 1989, pp. 51–60.

[56] L. Eshelman, R. A. Caruana, and J. Schaffer, "Biases in the crossover landscape," in *Proceedings of the third international conference on Genetic algorithms*, 1989, pp. 10–19.

[57] J. D. Schaffer and L. J. Eshelman, "On crossover as an evolutionarily viable strategy," in *International Conference on Genetic Algorithms (ICGA)*, 1991, pp. 61–68.

[58] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1992, pp. 1–37.

[59] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, 1995.

[60] R. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[61] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2-3, pp. 95–99, 1988.

[62] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial Intelligence*, vol. 40, no. 1-3, pp. 235–282, 1989.

[63] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[64] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, vol. 60, no. 2, pp. 139–144, 1988.

[65] D. B. Fogel, L. J. Fogel, and V. W. Porto, "Evolving neural networks," *Biological Cybernetics*, vol. 63, no. 6, pp. 487–493, 1990.

[66] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1994.

[67] D. B. Fogel, Ed., *Evolutionary Computation: A Fossil Record*. Wiley-Blackwell, 1998.

[68] K. Chellapilla and D. B. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1471–1496, 1999.

[69] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan Ann Arbor, 1975.

[70] ——, "Genetic algorithms: A 10 year perspective," in *Proceedings of an International Conference on Genetic Algorithms and their applications*, 1985.

[71] ——, "Learning with genetic algorithms," *Machine Learning*, vol. 3, no. 3, pp. 121–138, 1988.

[72] ——, *Evolutionary Computation: A Unified Approach*. MIT Press, 2016.

[73] R. M. Friedberg, "A learning machine: Part I," *IBM Journal of Research and Development*, vol. 2, no. 1, pp. 2–13, 1958.

[74] R. M. Friedberg, B. Dunham, and J. H. North, "A learning machine: Part II," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 282–287, 1959.

[75] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[76] ——, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[77] I. Rechenberg, "Cybernetic solution path of an experimental problem," in *Royal Air Force Establishment, Farnborough, Hampshire, England 1122*, 1965.

[78] H. P. Schwefel, *Understanding evolution as a collective strategy for groping in the dark*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 388–397.

[79] ——, "Evolutionary computation - History, status, and perspectives," in *Artificial Neural Networks - ICANN 96, (LNCS 112)*, 1996, pp. 15–15.

[80] H. Beyer and H. Schwefel, "Evolution strategies - A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[81] M. Conrad and H. H. Pattee, "Evolution experiments with an artificial ecosystem," *Journal of Theoretical Biology*, vol. 28, pp. 393–409, 1970.

[82] D. Fogel, R. W. Anderson, R. G. Reynolds, and W. Rizki, "Memorial tribute to Dr. Michael Conrad," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 1–2, 2001.

[83] R. R. Kampfner and M. Conrad, "Computational modeling of evolutionary learning processes in the brain," *Bulletin of Mathematical Biology*, vol. 45, no. 6, pp. 931–968, 1983.

[84] G. E. P. Box and N. Draper, *Evolutionary Operation. A Method for Increasing Industrial Productivity*. New York: Wiley, 1969.

[85] A. S. Fraser, "Monte Carlo analyses of genetic models," *Nature*, vol. 181, pp. 208–209, 1958.

[86] ——, "Simulation of genetic systems," *Journal of Theoretical Biology*, vol. 2, pp. 329–246, 1962.

[87] D. Fogel, "In memoriam Alex S. Fraser," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 429–430, 2002.

[88] L. J. Fogel, "A new concept: The kinalog system," *Journal of the Human Factors Society*, vol. 1, no. 2, pp. 30–37, 1959.

[89] ——, "Autonomous automata," *Industrial Research Magazine*, vol. 4, no. 2, pp. 14–19, 1962.

[90] V. Piuri, "In memoriam - Dr. Lawrence J. Fogel," *IEEE Computational Intelligence Magazine*, vol. 3, no. 1, pp. 68–69, 2008.

[91] J. Kietzmann, L. Pitt, and P. Berthon, "Disruptions, decisions, and destinations: Enter the age of 3-D printing and additive manufacturing," *Business Horizons*, vol. 58, pp. 209–215, 2015.

[92] B. Berman, "3-D printing: The new industrial revolution," *Business Horizons*, vol. 55, pp. 155–162, 2012.

[93] J. R. Koza, "Hierarchical genetic algorithms operating on populations of computer programs," in *IJCAI'89 Proceedings of the 11th International Joint Conference on Artificial intelligence - Volume 1*, 1989, pp. 768–774.

[94] ——, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[95] J. R. Koza and J. P. Rice, "Automatic programming of robots using genetic programming," in *AAAI'92 Proceedings of the tenth national conference on Artificial intelligence*, 1992, pp. 768–774.

[96] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk (with contributions from J. R. Koza), 2008.

[97] E. T. Barr, M. Harman, Y. Jia, A. Marginean, and J. Petke, "Automated software transplantation," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM, 2015, pp. 257–269.

[98] R. Harper, R. J. Chapman, C. Ferrie, C. Granade, R. Kueng, D. Naoumenko, S. T. Flammia, and A. Peruzzo, "Explaining quantum correlations through evolution of causal models," *Phys. Rev. A*, vol. 95, p. 042120, 2017.

[99] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, 2015.

[100] O. E. David, H. J. van den Herik, M. Koppel, and N. S. Netanyahu, "Genetic algorithms for evolving computer chess programs," *IEEE*

Transactions on Evolutionary Computation, vol. 18, no. 5, pp. 779–789, 2014.

[101] A. Elyasaf, A. Hauptman, and M. Sipper, "Evolutionary design of FreeCell solvers," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 4, pp. 270–281, 2012.

[102] D. Izzo, L. F. Simões, M. Märtens, G. C. de Croon, A. Heritier, and C. H. Yam, "Search for a grand tour of the jupiter galilean moons," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2013, pp. 1301–1308.

[103] C. Browne, *Evolutionary Game Design*. London: Springer London, 2011, ch. Yavalath, pp. 75–85.

[104] A. Elyasaf, A. Hauptman, and M. Sipper, "GA-FreeCell: Evolving solvers for the game of FreeCell," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2011, pp. 1931–1938.

[105] P. Widera, J. M. Garibaldi, and N. Krasnogor, "Evolutionary design of the energy function for protein structure prediction," in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 1305–1312.

[106] ——, "GP challenge: Evolving energy function for protein structure prediction," *Genetic Programming and Evolvable Machines*, vol. 11, no. 1, pp. 61–88, 2010.

[107] P. Widera, "Automated design of energy functions for protein structure prediction by means of genetic programming and improved structure similarity assessment," Ph.D. dissertation, School of Computer Science, University of Nottingham, UK, 2010.

[108] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, "Automatically finding patches using genetic programming," in *Proceedings of the 31st International Conference on Software Engineering*, 2009, pp. 364–374.

[109] S. Forrest, T. Nguyen, W. Weimer, and C. Le Goues, "A genetic programming approach to automated software repair," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2009, pp. 947–954.

[110] L. Spector, D. M. Clark, I. Lindsay, B. Barr, and J. Klein, "Genetic programming for finite algebras," in *Proceedings of the 10th Genetic and Evolutionary Computation Conference 2008*. ACM Press, 2008, pp. 1291–1298.

[111] S. Manos, M. C. J. Large, and L. Poladian, "Evolutionary design of single-mode microstructured polymer optical fibres using an artificial embryogeny representation," in *Proceedings of the 9th Genetic and Evolutionary Computation Conference 2007*. ACM Press, 2007, pp. 2549–2556.

[112] S. Kilinç, V. Jain, V. Aggarwal, and U. Cam, "Catalogue of variable frequency and single-resistance-controlled oscillators employing a single differential difference complementary current conveyor," *Frequenz*, vol. 60, no. 7-8, pp. 142–146, 2006.

[113] V. Aggarwal, "Novel canonic current mode DDCC based SRCO synthesized using a genetic algorithm," *Analog Integrated Circuits and Signal Processing*, vol. 40, no. 1, pp. 83–85, 2004.

[114] ——, "Evolving sinusoidal oscillators using genetic algorithms," in *Proceedings of 2003 NASA/DoD Conference on Evolvable Hardware*, 2003, pp. 67–76.

[115] S. Preble, M. Lipson, and H. Lipson, "Two-dimensional photonic crystals designed by evolutionary algorithms," *Applied Physics Letters*, vol. 86, no. 6, p. 061111, 2005.

[116] R. A. Bartels, M. M. Murnane, H. C. Kapteyn, I. Christov, and H. Rabitz, "Learning from learning algorithms: Application to attosecond dynamics of high-harmonic generation," *Phys. Rev. A*, vol. 70, p. 043404, 2004.

[117] R. Bartels, S. Backus, E. Zeek, L. Misoguti, G. Vdovin, I. P. C. an M. M. Murnane, and H. C. Kapteyn, "Shaped-pulse optimization of coherent emission of high-harmonic soft x-rays," *Nature*, vol. 406, pp. 164–166.

[118] J. D. Lohn, G. S. Hornby, and D. S. Linden, *An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission*. Boston, MA: Springer US, 2005, pp. 301–315.

[119] L. Spector, *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Springer US, 2007.

[120] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[121] E. K. Burke, M. Hyde, G. Kendall, and J. Woodward, "Automating the packing heuristic design process with genetic programming," *Evolutionary Computation*, vol. 20, no. 1, pp. 63–89, 2012.

[122] E. K. Burke, M. Hyde, and G. Kendall, "Grammatical evolution of local search heuristics," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 406–417, 2012.

[123] P. Cowling, G. Kendall, and E. Soubeiga, *A Hyperheuristic Approach to Scheduling a Sales Summit*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 176–190.

[124] J. Denzinger, M. Fuchs, and M. Fuchs, "High performance ATP systems by combining several AI methods," Technical Report, SEKI-Report SR-96-09, University of Kaiserslautern, Tech. Rep., 1996.

[125] H. Fisher and G. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling*, J. Muth and G. Thompson, Eds. Prentice Hall, 1963, pp. 225–251.

[126] W. B. Crowston, F. Glover, G. Thompson, and J. D. Trawick, "Probabilistic and parametric learning combinations of local job shop scheduling rules," ONR Research memorandum, GSIA, Carnegie Mellon University, Pittsburgh 1(117), 1963.

[127] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[128] S. Asta, E. Özcan, and A. J. Parkes, "Champ: Creating heuristics via many parameters for online bin packing," *Expert Systems with Applications*, vol. 63, pp. 208–221, 2016.

[129] N. R. Sabar, M. A. G. Kendall, and R. Qu, "A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 217–228, 2015.

[130] ——, "Automatic design of hyper-heuristic framework with gene expression programming for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 309–325, 2015.

[131] S. Rosenberg, *Dreaming in Code: Two Dozen Programmers, Three Years, 4,732 Bugs, and One Quest for Transcendent Software*. Three Rivers Press (CA), 2008.

[132] F. P. Brooks Jr., *The Mythical Man-Month*. Addison-Wesley, 1975.

[133] C. G. Johnson, "Fitness in evolutionary art and music: A taxonomy and future prospects," *International Journal of Arts and Technology*, vol. 9, no. 1, pp. 4–25, 2016.

[134] S. Todd and W. Latham, Eds., *Evolutionary Art and Computers*. Academic Press Inc, 1992.

[135] M. Harman and B. F. Jones, "Search-based software engineering," *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, 2001.

[136] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall, "A tabu search hyper-heuristic strategy for t-way test suite generation," *Applied Soft Computing*, vol. 44, pp. 57–74, 2016.

[137] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren, "Adaptive multi-objective evolutionary algorithms for overtime planning in software projects," *IEEE Transactions on Software Engineering*, In Press.

[138] K. Sörensen, "Metaheuristics – the metaphor exposed," *International Transactions in Operational Research*, vol. 22, no. 1, pp. 3–18, 2015.