Quo Vadis, Evolutionary Computation?
On a Growing Gap between Theory and Practice

Zbigniew Michalewicz*

School of Computer Science,
University of Adelaide,
Adelaide, SA 5005, Australia
zbyszek@cs.adelaide.edu.au

Abstract. At the Workshop on Evolutionary Algorithms, organized by the Institute for Mathematics and Its Applications, University of Minnesota, Minneapolis, Minnesota, October 21 – 25, 1996, one of the invited speakers, Dave Davis made an interesting claim. As the most recognised practitioner of Evolutionary Algorithms at that time he said that all theoretical results in the area of Evolutionary Algorithms were of no use to him – actually, his claim was a bit stronger. He said that if a theoretical result indicated that, say, the best value of some parameter was such-and-such, he would never use the recommended value in any real-world implementation of an evolutionary algorithm! Clearly, there was – in his opinion – a significant gap between theory and practice of Evolutionary Algorithms.

Fifteen years later, it is worthwhile revisiting this claim and to answer some questions; these include: What are the practical contributions coming from the theory of Evolutionary Algorithms? Did we manage to close the gap between the theory and practice? How do Evolutionary Algorithms compare with Operation Research methods in real-world applications? Why do so few papers on Evolutionary Algorithms describe real-world applications? For what type of problems are Evolutionary Algorithms "the best" method? In this article, I'll attempt to answer these questions – or at least to provide my personal perspective on these issues.

1 Inspiration

Since the publication of my first book on Genetic Algorithms (Michalewicz, 1992) exactly twenty years ago, I have been thinking about theory and practice of these algorithms. From my early experiments it was clear that it would be necessary to extend the binary representation of genetic algorithms by some other data structures (hence the 'data structure' term in the title of the book) and incorporate problem-specific knowledge into the algorithm. These initial thoughts were later supplemented by various research activities as well as real world experiences – working on many

* Also at the Institute of Computer Science, Polish Academy of Sciences and at the Polish Japanese Institute of Information Technology, Warsaw, Poland, and the Chairman of the Board at SolveIT Software Pty Ltd.

J. Liu et al. (Eds.): WCCI 2012 Plenary/Invited Lectures, LNCS 7311, pp. 98–121, 2012.

projects for very large companies (e.g. Ford, General Motors, BMA Coal, Orlando Wines, Bank of America, BHP Billiton, Viterra. Dentsu, Rio Tinto, Xstrata, Fortescue). This chapter summarizes my dual experience – academia vs. industry – and I hope it would be useful for the current and the future generations of researchers in the Evolutionary Computation (EC) community. After all, it took me 20 years sitting on both sides of the fence to collect my thoughts!

However, the inspiration for this chapter (and my talk at the IEEE CEC'12) came from reading a relatively recent short article by Jeff Ullman on how to advise PhD students (Ullman, 2009). By the way, my PhD (from 1981) was in database systems (at that time I have not even heard about evolutionary/genetic algorithms) and Jeff Ullman was one of my heroes – his articles and books at that time set directions for (usually theoretical) research in many areas of database management – from database design to concurrency control (of course, he has made also substantial contributions in other areas of computer science). At that time I considered him as one of the leading researchers in theoretical computer science – it is why I found his thoughts (written up recently – six years or so after his retirement) a bit surprising!

In his article he addressed the issue of advising PhD students, and many comments he made were applicable to a much wider audience. For example, he discussed a standard way to write and publish a paper: "*Look at the last section [of some paper], where there were always some 'open problems.' Pick one, and work on it, until you are able to make a little progress. Then write a paper of your own about your progress, and don't forget to include an 'open problems' section, where you put in everything you were unable to do*." Indeed, it is hard to disagree – and this is what we often experience reading papers written by researchers from the Evolutionary Computation community… For example, one researcher proposes a new method for something and the method includes a few parameters – the follow-up researcher tunes these parameters and the next researcher proposes adaptive method for handling them. The number of examples of such papers (whether experimental or theoretical) is extensive.

However, Jeff Ullman does not believe that such approach is a good one: "*Unfortunately this approach, still widely practiced today, encourages mediocrity. It gives the illusion that research is about making small increments to someone else's work. But worse, it almost guarantees that after a while, the work is driven by what* ***can*** *be solved, rather than what* ***needs*** *to be solved. People write papers, and the papers get accepted because they are reviewed by the people who wrote the papers being improved incrementally, but the influence beyond the world of paper-writing is minimal*."

This is probably a fair introduction and a summary of this chapter – I will argue that the gap between the theory and practice of Evolutionary Algorithms (EA) is getting wider, partially because of reasons identified by Jeff Ullman, and partially because complexity of business problems increased many times over the last 15 years (mainly due to the globalization and integration processes)[1], whereas the theory still studies the same types of problems (TSP, JSSP, Knapsack Problem, sphere

[1] Scott Wooldridge, Vice President Industry Business at Schneider Electric Australia said (private communication): "The world is becoming more dynamic and more interconnected each year. Managers and executives are finding that changes in supply and demand can be sudden and unexpected. The complexity of doing business is increasing at a rapid rate and it is becoming hard to anticipate events, interpret their potential impact and understand the implication of various decisions at both an operational and strategic level."

function, etc.). Indeed, the influence of many research papers in the EC community, beyond the world of paper-writing, is minimal. Of course, the problem is much broader – note that publication quantity is often used as a proxy of academic success and influences funding decisions, however the research questions that need answering don't always lend themselves to rapid incremental publications. Moreover, there is no substantial reward for research that demonstrates linkage with the real world, thus no real incentive for academia to solve the problems that industry is really interested in.

His further thought was the following: "*In the first years of computer science as an academic discipline, many theses were 'theoretical,' in the sense that the contribution was mostly pencil-and-paper: theorems, algorithms and the like, rather than software. While much of this work was vulnerable to the problem just described – paper building on paper – it is quite possible for a theoretical thesis to offer a real contribution*." Then he discussed as a case from many years ago where a "theoretical" piece of work was not based on what some paper left open, but rather on an expressed need of the real-world – and it was a great success in the real-world environment. Another example given was that of Sergey Brin and Larry Page, who saw the need for a better search engine and the key ways that goal could be reached. Clearly, "*…there needs to be an exposure to problems that are at the frontier, and that are needed by a 'customer.' Sometimes, they can find a customer in industry […]. Summer internships can be a great opportunity. However, advisors should encourage students to intern at a strong industrial research group, one where the goals are more than minor tweaks to what exists. Whether the thesis is theoretical or an implemented solution, students need to be guided to understand who will consume their contribution if they are successful. And the answer cannot be 'people will read the paper I will write, and they will use the open problems in it to help form their own theses.' Especially when dealing with theoretical theses, the chain of consumption may be long, with idea feeding idea, until the payload is delivered. Yet if we let students ignore the question of whether such a chain and payload plausibly exist, we are doing them a disservice*."

The EC community should be immune from the problem of identifying possible payloads as evolutionary algorithms are directly applicable to a variety of real-world problems. Indeed, many EA papers indicate a strong connection between the presented approach and real-world applicability. But there is a huge difference between a "strong connection" and an "implementation" (or practice) – and it would be necessary for the EC community to make this transition sooner than later. Otherwise, EAs would be perceived as one of many methods one can just try in some special circumstances – whereas these algorithms have a potential to deliver a significant return on investment for many hard real-world problems (see further sections of this chapter).

To progress our discussion on the gap between theory and practice of evolutionary algorithms, first we have to address two fundamental questions: (1) what is an evolutionary algorithm? and (2) what is practice (i.e. a real-world application)? These two questions are important because without some discussion of them it would be difficult to discuss the current gap[2] between their theory and practice.

[2] A continuing gap between theory and practice can be viewed as a positive feature of any dynamic, exciting, growing field, however, it seems that in the case of Evolutionary Computation this gap is too large…

For example (this emerged during my recent private correspondence with Hans-Georg Beyer, the current editor-in-chief of *Evolutionary Computation Journal*) some researchers believe that for operational-type problems (i.e. problems that require optimisation with some regular frequency) "*…you may do better by design a problem-specific optimisation algorithm. And this will be quite often not an EA due to different reasons.*" However, the question is how to distinguish between "a problem-specific optimisation algorithm" and "an EA with problem-specific variation operators and some other non-standard features"?

So these two topics are discussed in the following two sections of this chapter (sections 2 and 3), whereas section 4 talks about the gap between theory and practice. Section 5 concludes the chapter.

2 What Is an Evolutionary Algorithm?

The field of meta-heuristics has a rich history. Many meta-heuristics have emerged during the past 30 years; many of them have been inspired by some aspects of nature, ranging from the cooling of metal to the movements of ants. Meta-heuristics methods include a variety of hill climbing techniques (deterministic and stochastic), ant colonies, artificial immune systems, differential evolution, particle swarms, simulated annealing, tabu search, cultural algorithms, evolutionary and co-evolutionary algorithms. These meta-heuristics can be classified into some categories based on different criteria. For example, some meta-heuristics process single solution (e.g. simulated annealing) whereas some others process a set of solutions (and are called population-based methods, e.g. evolutionary algorithms). Some meta-heuristics can be deterministic (e.g. tabu search), some other are stochastic (e.g. simulated annealing). Some meta-heuristic generate complete solutions by *modifying* complete solutions (e.g. evolutionary algorithms), whereas some other *construct* new solutions at every iteration (e.g. ant systems). Many of these meta-heuristics offer some unique features (e.g. use of memory, use of 'temperature', use of methods for exchange information between individuals in population-based methods). Further, even within a single meta-heuristic, there are many variants which incorporate different representations of solutions and different operators for generating new solutions.

However, there is one common denominator: all meta-heuristics strive to create high quality solutions by making a series of improvements during their iterative process. Whether they start with (randomly generated) low quality solutions or they use smart initialisation methods to take advantage of the problem-specific knowledge, they aim to improve solution quality during the search process. At any iteration a meta-heuristic method must make some 'selection' decisions: which solutions of the current iteration should be kept for further processing and which should be discarded? And this selection step is quite important as it is often responsible for maintaining balance between exploration and exploitation of the search space – e.g. strong selective pressure harms exploratory capabilities of the algorithm and often results in a premature convergence. Of course, different meta-heuristics address this question in

their own way. For example, evolutionary algorithms usually split selection process into two independent activities: (1) selection of parents and (2) selection of survivors.

So, what is an evolutionary algorithm and how does it differ from other meta-heuristics? Well, this is actually a very good question! Of course, one can explain evolutionary algorithms as the ones based on concepts of natural evolution. It is also possible to discuss 'mutation' and 'crossover' operators, parents and offspring, not to mention the Darwinian principle of natural selection and survival of the fittest. However, from a high-level perspective things are not so clear. Say, you are solving a new challenging optimization problem and you have designed a new method. So you have selected some appropriate representation for candidate solutions for the problem and agreed on the evaluation function which would measure the quality of solutions. Further, you came with (more or less clever) heuristic method for finding a few initial solutions, and (after analysing the characteristics of the problem) you have also designed a few variation operators which would be responsible for modifying the current solutions (thus creating new set of candidate solutions). You have also incorporated some simple selection method (say, ranking method, where the better individual has better chances to be selected) at any iteration. You have extended the system by some additional non-standard features (e.g. special repair method or decoder to deal with problem-specific constraints). Finally, you experimented with the system and as the result you have tuned a few parameters of the method (e.g. population size, rates of operators).

Have you just created an evolutionary algorithm? Note that if your answer is 'yes', meaning “yes, I have created a variant of an evolutionary algorithm”, the consequence might be that any meta-heuristic method which searches for a solution in iterative manner can be labelled as a variant of 'evolutionary algorithm'. For example, simulated annealing can be considered as a variant of (1 + 1) evolutionary algorithm with an adaptive selection method; tabu search can be considered as (1 + 1) evolutionary algorithm with memory-based selection method. In general, there are many iterative stochastic search methods – are all of these ‘evolutionary algorithms’? Recall also, that some ‘evolutionary algorithms’ have been 'extended' by memory structures (e.g. when they operate in dynamic environments or to keep the search history – Chow & Yuen, 2011) or by a parameter called 'temperature' (to control mutation rates). And it is possible to provide many other examples!

Consider, for example, the Newton method – the method for finding successively better approximations to the roots of a real-valued function. For one-dimensional functions, the Newton method generates an initial individual x_0 and generate the next point x_1 (provided the function is reasonably well-behaved):

$$x_1 = x_0 - f(x_0)/f'(x_0)$$

where $f'(x)$ is derivative of $f(x)$. Geometrically, x_1 is the intersection with the x-axis of a line tangential to f at $f(x_0)$. The process is repeated:

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

until a sufficiently accurate (i.e. near-optimum) value is reached. So the method works as follows: one starts with an initial guess which is reasonably close to the true root (i.e. intelligent initialisation), then a variation operator (problem-specific mutation) is applied to generate offspring. The selection method always selects offspring for further processing. These steps are repeated for several iterations (we should say: generations), until a termination condition is met (the error drops below a predefined threshold).

Did Newton discover an 'evolutionary algorithm'? Before you answer, recall that (1 + 1) evolutionary strategy does not require population of solutions as it is processing just one individual (which is compared with its only offspring). Recall, that many 'evolutionary algorithms' assume a deterministic selection method; many 'evolutionary algorithms' take advantage of smart initialisation and problem-specific operators. Yes, I realise that stochastic component is missing, but it would be relatively easy to modify the method from deterministic one to stochastic. For example, we can easily modify the formula for generating the 'offspring':

$$x_{n+1} = x_n - f(x_n)/f'(x_n) + N(0,\delta)$$

Note also that the argument that evolutionary algorithms in numerical domains are derivative-free (hence applicable to discontinuous functions) will not hold here as there have been many evolutionary algorithms proposed with problem-specific variation operators, which take advantage from the knowledge of the shape of the landscape. Not to mention that – as for evolutionary algorithms – we can test the modified Newton method on different landscapes, investigate its convergence rates, investigate its scalability for higher dimensions, investigate the issue of deceptiveness, etc.

Newton's method emerged in 17th century, however, his method was probably derived from a similar but less precise method by Vieta. And the essence of Vieta's method can be found in the work of the Persian mathematician, Sharaf al-Din al-Tusi. So we can rewrite the history of evolutionary algorithms moving their roots centuries earlier as there are many examples of search methods based on (smart) initialisation, (problem-specific) mutation, selection, iterations (generations), termination conditions… It seems that the era of evolutionary algorithms[3] started with just a new terminology – all pioneers of EAs have been using 'right' vocabulary – generations (instead of iterations), mutation and/or crossover (instead of just 'variation operator'), the Darwinian selection – survival of the fittest (instead of just selection), etc. So what does it take to call an iterative search algorithm – an 'evolutionary algorithm'? Terminology used? A stochastic component? Or something else?

These remarks are also applicable to the whole field of modern heuristic methods. Over the last few years we saw emergence of a variety of new methods; these include bee colony optimisation, honey-bee mating optimisation, glow-worm swarm

[3] I do believe that the *modern* era of evolutionary algorithms started at the 4th International Conference on Genetic Algorithms (San Diego, 1991) when Evolution Strategies and Genetic Programming approaches were presented to the participants of the conference, together with other approaches based on different data structures (like the original Evolutionary Programming approach).

optimisation, intelligent water drops, firefly algorithm, the monkey search, cuckoo search, galaxy-based search algorithms, spiral optimisation… Which of these are evolutionary algorithms? For example, in cuckoo search each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests (in the simplest form, each nest has one egg only and the algorithm can be extended to more complicated cases in which each nest has multiple eggs representing a set of solutions). Cuckoo search is based on three idealized rules: (1) each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest; (2) the best nests with high quality of eggs will carry over to the next generation; and (3) the number of available hosts nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with some probability. Clearly, it would be easy to rewrite all these in 'evolutionary' terminology (e.g. including the island population model)…

This very issue was recently addressed by Fred Glover[4] in his article *Sex and Metaheuristic Metaphors,* where a 'new' meta-heuristic method, the *Courtship Algorithm*, is proposed. The algorithm is loaded with some principles that people have applied to courtship – and these principles translate directly into rules for meta-heuristic methods. These include: (a) *Have we met before*? (i.e. use memory to exploit recurrences), (b) *Your place or mine*? (i.e. consider the benefits of regional exploration), (c) *Variety is spice* (i.e. employ diversification at all levels), (d) *Once is not enough* (i.e. iterate over good options), (e) *No need to be timid* (i.e. well-timed aggressive moves can pay off), or (f) *Don't stop now*! (i.e. take advantage of momentum). Of course, they are many more of such useful principles one can use – reading this article I recalled that in 1999 I published a paper (Hinterding et al. 1999) titled *Your Brains and My Beauty: Parent Matching for Constrained Optimisation* – very much in line with the Courtship Algorithm! So, is Evolutionary Algorithm a special case of Courtship Algorithm or vice versa?

It seems that many researchers are just playing games with terminology. In particular, it is always possible to extend the definition of something to include everything else, and people often do this, although it obviously doesn't establish any real generality to the "extended definition". Moreover, the architecture of so-called "special cases" may be more intricate and advanced than the architecture that has been constructed for the "general case" (as where a general-case design of a building may have nothing of the complexity and intricacy that must go into the design of a skyscraper). A similar sort of thing also occurs in speaking of probabilistic and deterministic methods. We can say that deterministic methods are a special case of probabilistic methods, because deterministic methods result by making all probabilities 1 and 0. But this blurs a great deal of the special essence of each category of method.

It seems that Evolutionary Algorithms, in the broad sense of this term, provide just a general framework on how to approach complex problems. All their components, from the initialisation, through variation operators and selection methods, to constraint-handling methods, might be problem-specific. Some researchers (but not that many, I believe) in the Evolutionary Computation community agree – and view an

[4] See `http://optimaldecisionanalytics.typepad.com/`

EA as a meta-heuristic in the broad sense of the term by providing a framework for creating/instantiating problem specific heuristic methods [De Jong, 2002]. Evolutionary Algorithms, to be successfully applied, must be tailored to a specific domain in order to be useful and provide decent results in specified time; without any doubt, incorporating domain knowledge leads to more effective EA searches. Probably this is why so many authors talk about 'variants', 'modified versions', 'extensions', or just 'knowledge-enhanced' evolutionary algorithms in their application-oriented papers – for which theoretical results usually are not relevant.

3 What Is 'Practice'? What Is a Real-World Application?

Clearly, the 'practice' of evolutionary algorithms is connected with real-world applications – after all, this is what the term 'practice' means (at least in the context of evolutionary algorithms). But as it was the case with the definition of evolutionary algorithms, the meaning of the term 'real-world application' is not that clear at all!

The Evolutionary Computation community over the last 30 years has been making claims that their methods are ideal for hard problems – problems, where other methods usually fail. As most real-world problems are very hard and complex, with nonlinearities and discontinuities, complex constraints and business rules, noise and uncertainty, EAs should provide a great benefit to the real-world community.

Even today, many research papers point to ability of evolutionary algorithms to solve real-world problems. For example, I reviewed the last two issues (Spring and Summer 2011) of the *Evolutionary Computation Journal* and the last two issues (October and December 2011) of the *IEEE Transactions on Evolutionary Computation*. In the introductory paragraphs of some papers included in these issues I found the following sentences:

- "*Evolutionary algorithms are a wide class of solution methods that have been successfully applied to many optimisation problems*,"
- "*Despite the lack of theoretical foundation, simplicity of Evolutionary Algorithms has attracted many researchers and practitioners*,"
- "*EAs are randomized search heuristics that solve problems successfully in many cases*,"
- "*…but in practice they optimise many challenging problems effectively*,"
- "*Randomized search algorithms have been very successful is solving combinatorial optimisation problems,"*
- *"For the past three decades, many population based search techniques have surfaced to become a mainstay of optimisation*,"
- "*However, as many real-world optimisation problems are black-box problems of which* a priori *problem knowledge is not available, the use of metaheuristics started to prevail*,"
- "*EAs are used for solving many various problems*," or
- "*Recently, researchers are increasingly directing their focus on solving multi-objective optimisation problems (MOPs), since the solution of these problems is of great importance in the areas of economics, engineering, biology, medicine, materials, and so on*."

Some papers focus on classic problems (e.g. graph partitioning, maximum clique) – and in such cases the authors make a connection with real world:

- "*Graph partitioning is one of the fundamental combinatorial optimisation problems which is notable for its applicability to a wide range of domains, such as very large scale integration (VLSI) design, data mining, image segmentation, and so on,*" and
- "*The maximum clique (MC) problem in graphs is a paradigmatic combinatorial optimisation problem with many relevant applications, including information retrieval, computer vision, and social network analysis*."

The influence of benchmark problems on the EA research manifests itself in statements like "*When genetic algorithms (GAs) are applied to combinatorial problems, permutation representation is usually adopted*" which is very true for many benchmark problems and very false for real-world problems.

It is my opinion that these sentences speak loud and clear about desires of many researchers to see a stronger connection between their research and the real-world (or their belief that this is the case!). However, these statements are harmful for the community as they give a misleading perspective on the current spread of evolutionary methods in businesses and industries and provide unjustified psychological comfort. Where are all these applications that the authors refer to? Where do you see the prevalence of EAs in business and industry? When did these techniques become the 'mainstay of optimisation'? Where are all these success stories? What does it mean that EAs have been successfully applied to many optimisation problems? In my recent essay (Michalewicz, 2012) I addressed this very issue and concluded that (for many reasons) there are not that many real world implementations of EAs after all…

Further, it seems that many researchers are interpreting the phrase "real-world application" in a quite arbitrary way. I have looked through proceedings of many international conferences on evolutionary algorithms (e.g. GECCO, IEEE CEC, PPSN) which often include special sessions on "real-world applications". I have checked many journals (e.g. *Journal of Evolutionary Algorithms*, *IEEE Transactions on Evolutionary Computation*) with special attention to their special issues on real-world applications (e.g. scheduling) and a few edited volumes which aimed at real-world applications. The main problem is that all these "real-world applications" look like "Guinea pigs" – not really from Guinea, and not really pigs... It seems that researchers are interpreting the phrase ""real-world applications" in very arbitrary way. However, it is relatively easy to group all these "real-world applications" into four broad categories:

1. Applications used in some business/industry on daily (regular) basis.
2. Applications tested on real data (e.g. taken from a hospital, city council, a particular business unit).
3. Applications tested on some well-known model (e.g. TSP, VRP, JSSP) of a real-world problem.
4. Other applications (e.g. numerical optimisation, constraint-handling, multi-objective optimisation).

Of course, this classification is not that precise – one can imagine a border case between categories 1 and 2, where an application might be used intermittently, e.g. in some relatively infrequent planning or modelling task. However, the question is: Are the applications in all these four categories real-world applications? Clearly, category 1 is beyond any dispute. But the case of category 2 is not that clear – it is true that real-world data were used in experimentation; however, application as such is not real-world application – as no one uses it. It is rather an indication, that such approach – in the real-world setting – may have a merit. Category 3 is also muddy – there is no question that some models, e.g. for vehicle routing, scheduling, distribution, are models of real-world environments, however, these applications do not operate in real-world environment (and most likely, they never will). And most other novel applications (category 4), whether they aim at handling constraints, many objectives, or the issue of noise – while very important and directly applicable for real-world settings, are hardly real-world applications.

Clearly, there are also EA-based "solvers" and a variety of EA-based "tools" available; however, there is a huge difference between these and real-world software applications. Solvers and tools are sometimes useful, but they have very limited applicability and a limited audience. Some individuals in some companies use solvers which are based on modern heuristic methods (and Evolutionary Algorithms in particular) for solving some small-scale optimisation problems; however, due to the limited usefulness of such tools I heard quite often negative statements from various researchers employed in major companies, e.g. "*I tried a Genetic Algorithm on my problem, but I did not get any results…They do not work*".

Some senior researchers agree. Garry Greenwood, the current editor-in-chief of the *IEEE Transactions on Evolutionary Computation*, wrote recently:[5] "*I believe the MATLAB GA toolbox has been devastating to the EC field and has done far, far more damage than good. I wish it had never been created. It was designed to do searches over an extremely broad class of optimisation problems. No domain knowledge could be incorporated so the search results on non-toy problems were often mediocre. More than one person has told me he tried a GA using that MATLAB toolbox and wasn't impressed with the results. Hence, he was convinced GAs weren't very good. The EC community has taken a bad rap for years because people formed a negative opinion of GAs based on one encounter with that damn MATLAB toolbox*."

There are, of course, papers describing some interesting applications (e.g. applications of Covariance Matrix Adaptation Evolution Strategies – CMA-ES[6]), however, many of these are not real-world applications, they are usually restricted to continuous variables only, and they hardly address complex business problems. There are also many papers which aim at classic Operation Research (OR) problems, e.g. traveling salesman problems, job shop scheduling problems, graph colouring problems, variety of vehicle routing problems, knapsack problems, packing and cutting problems – but all these have very limited significance for today's real-world problems.

[5] Private correspondence.

[6] See `http://www.lri.fr/~hansen/cmaapplications.pdf`

Let's illustrate the point of expectations of the real-world community in the context of supply chain – understood as "…a combination of processes, functions, activities, relationships and pathways along which products, services, information and financial transactions move in and between enterprises, in both directions (Gattorna, 2010). One can visualise a supply chain as a combination of multiple logistic networks which involve many upstream and downstream organisations – and clearly, the potential of significant improvement in performance of such network is much greater than within a single silo[7]. Note, for example, that the issue of scheduling production lines (e.g. maximising the efficiency, minimizing the cost) has direct relationships with inventory and stock-safety levels, replenishments strategies, transportation costs, deliver-in-full-on-time, to name a few. Moreover, optimising one silo of the operation may have negative impact on upstream and/or downstream silos[8]. Thus businesses these days need "global solutions" for their whole operation, not silo solutions. This was recognised over 30 years ago by Operations Research community; in 1979 R. Ackoff wrote: "*Problems require holistic treatment. They cannot be treated effectively by decomposing them analytically into separate problems to which optimal solutions are sought*." Only an integrated view of supply chain operations would increase visibility and transparency across end-to-end supply chains. Managers these days are looking for applications that, for example, will enable fully integrated, financially-driven Sales & Operations Planning with, for example, transportation costs, working capital requirements, and stock outs as priorities. They need to create within such applications consistent what-if scenarios to facilitate the planning, scheduling, and optimisation of multi-site sourcing, production, storage, logistics, and distribution activities – all these in time changing (dynamic) environments.

It seems to me that (from a high level perspective) most real-world problems fall into two categories: (1) design/static problems, and (2) operational/dynamic problems. The first category includes a variety of hard problems, like TSP, VRP, JSSP, graph coloring, knapsack problem, and millions of others. Some of them are really hard – and I believe that 99% of research aims at these type of problems. I do believe, however, that the problems from the second category are (a) much harder, (b) the standard methods usually fail for them, and (c) they represent a huge opportunity for EAs. Let me explain further.

The systems to address problems in the 2nd category are really decision-support systems that require continuous flow of data, predictive components, almost immediate recommendations for recovering from sudden changes, etc. Not to mention that they should handle the 1st category instances as they have to solve static versions of the problem as well. This is altogether a different game, and the differences between problems in the 1st and 2nd categories are huge. Problems in the 2nd category are usually multi-silo problems as opposed to single silo for the 1st category problems. Problems in the 2nd category usually deal with many variables, nonlinear relationships, huge varieties of constraints (e.g. constraints in real-world settings often include

[7] We use the term 'silo' for one component of a supply chain.

[8] Of course, there are some real-world scientific and engineering problems where a good solution to a silo problem is beneficial (e.g. a variety of design problems), but they do not represent complexities of today's business problems.

'if-then' conditions), business rules, many (usually conflicting) objectives – and all of these are set in a dynamic and noisy environment. These problems belong to an optimization class, as they require recommendations for "the best" decision at the moment (whichever complex characteristic of what "the best" may mean – often with risk factors included). An additional feature of these systems is that they do not operate on crisp values but rather on probability distributions. For example, one thing is to assume that the travel from one point to another takes 36 hours, another thing is to accept a probability distribution with mean 36 and standard deviation determined from past data. Not to mention that these probability distributions change over time (the system learns) as new data are coming in (e.g. due to some improvements, six months later the mean is 35.5 hours, and standard deviation is much smaller). Further, the response time is expected to be much shorter – applications for the design-type problems are not time-critical, whereas applications for operational-type problems are. Finally, robustness of a solution is as important as its quality... In comparison with such problems, NP-hard problems like TSP seem to be toy problems (which they are, in a sense).

Clearly, these are types of problems which should be addressed by Evolutionary Computation community, as they represent hard real-world problems where other methods usually fail.[9] These are the problems for which meta-heuristic methods in general (and EA-based methods in particular) should be the methods of choice.

4 Theory versus Practice

As indicated in the abstract of this chapter, at the Workshop on Evolutionary Algorithms, organized by the Institute for Mathematics and Its Applications, University of Minnesota, Minneapolis, Minnesota, October 21 – 25, 1996, one of the invited speakers, Dave Davis made a claim that all theoretical results in the area of Evolutionary Algorithms were of no use to a practitioner. At that time there was – in his opinion – a significant gap between theory and practice of Evolutionary Algorithms.

So where are we 15 years later with this issue? Did we manage to close the gap between the theory and practice? It seems that there is still a significant mismatch between the efforts of hundreds of researchers who have been making substantial contributions to the theory of evolutionary computation over the years and the number of real-world applications which are based on concepts of evolutionary algorithms – it seems also, that this gap is still growing.

I believe that there are two main reasons for this phenomenon; these are:

1. The growing complexity of real-world problems
2. The focus of research community on issues which are secondary for real-world applications

[9] A recent report (private correspondence) generated by several Operations Research experts on optimisation of a particular supply chain with 37 nodes and 69 arcs, 862,000 variables and 1.6 million constraints, would require 18 hours per objective by mixed integer programming.

The first point was already touched on in the previous section of the chapter. For many reasons today's business problems are of much higher complexity than 30 years ago and the real-world is searching for techniques which would address their problems – problems which are loaded with nonlinearities and/or discontinuities, many (possibly conflicting) objectives, variety of business rules, soft and hard constraints, and noise.

However, it seems to me that many researchers in the EC community are quite disconnected with today's business, and they still believe that[10]: "*In business applications, the goal function is often quite simple, a linear one. Most of the modelling is put into the inequality constraints which very often can be linearized and integer conditions can be relaxed. As a result, one often ends up with an LP*[11]*. And if not, then the modelling is changed in such a manner that the resulting problem is again an LP. This is the main reason why CPLEX (and some others) is so prevalent*."

I think that a perception that in business applications the goal function is often quite simple, was right 30 years ago, but today it is just wrong and there is nothing further from the truth. The real complexity exists mainly in real-world problems (and not in artificial benchmarks). The business goal functions usually are extremely complex, not to mention constraints and business rules, which are often conditional, calendarised, dynamic, etc. and they interact with many objectives in complex ways. And the main reasons of the popularity of LP methods are the ease of their use and the knowledge, how to use them – which is not the case with EAs. The main reason of their popularity is *not* the quality of results... Many business units dream about replacing their software which incorporates LP as their optimisation engine, but they do not see alternatives. There is also an amazing amount of inertia in the business that has nothing to do with what technology really works – and it is much easier to *plug-in* CPLEX than to *develop* an EA for a problem at hand…

Real-world optimisation problems involve large number of variables, numerous objectives, constraints, and business rules, all contributing in various ways to the quality of solutions. The complexity of such problems makes it virtually impossible for human domain experts to find an optimal solution. Further, manual adjustments of scenarios (what-if scenarios and trade-off analysis), which are needed for strategic planning, become an expensive or unaffordable exercise. The main reason behind this complexity is that large-scale business problems consist of several interconnected components, which makes many standard approaches ineffective. Even if we know exact and efficient algorithms for solving particular components or aspects of an overall problem, these algorithms only yield solutions to sub-problems, and it remains an open question how to integrate these partial solutions to obtain a global optimum for the whole problem. Moreover, optimising one silo of the operation may have negative impact on upstream and/or downstream silos. For example, at PPSN'10 I gave a keynote talk on a powerful EA-based enterprise[12] software application that was

[10] This is the exact quote from my private correspondence with one of senior members of the EC community.

[11] Linear Programming.

[12] Enterprise software addresses the needs of organization processes and data flow, often in a large distributed environment (e.g. supply-chain management software).

developed recently[13] to address many of wine production challenges present in different parts of the wine supply chain. This enterprise software application for wine industries consists of a suite of software modules (these include predictive modelling for grape maturity, using weather forecasts and readings on Baum, PH, and TA, vintage planning, crush scheduling, tank farm optimisation, bottling-line sequencing, and demand planning) that can optimise the end-to-end wine supply chain.. When deployed together, these applications can optimise all planning & scheduling activities across a winery's entire supply chain.

Each of these individual software application deals with a silo problem; for example, the bottling module (Mohais et al. 2011) is responsible for generating optimal production schedules for the wineries' bottling operations. Opportunities for optimisation include manipulating the sequencing order, selecting which bottling lines to use, consolidating similar orders within the planning horizon, and suggesting changes to the requested dates to improve the overall schedule. Some of the key objectives are to maximise export and domestic service levels (i.e. DIFOT), maximising production efficiency, and minimising cost. As the module provides decision-support, the user has full control over the optimisation process in that they are able to lock in manual decisions, set the business rules and constraints, re-optimise after making changes, and compare the current solution with an alternative plan. The module also provides a *what-if* tab that can be used to analyse strategic business decisions and events such as capital investment in new equipment, or to look at operational decisions like adding or removing extra shifts, or even for crisis management (what is the impact of a bottling line going down or key staff being ill). Reporting is provided on a number of levels and to suit different stakeholders; for example the daily bottling programs for execution, a report on particular wine blends, or a report on expected production efficiency. Users are also able to generate an alternative to the current production schedule, with the system providing a comparison to help the user evaluate the impact of any changes. The comparison includes performance metrics (KPI's) such as any difference in the number of late orders, and changes to production efficiency and cost (this could include measures such as cost per unit, total production throughput, production line utilisation, etc.). This allows the user to experiment with different schedules before committing to making any changes; for example, trying to incorporate a last minute export order without disrupting existing orders.

Each of these modules (whether predictive modelling for grape, vintage planning, crush scheduling, tank farm optimisation, bottling-line sequencing, and demand planning) represents a significant large-scale optimisation/prediction problem in itself and includes several interesting research aspects (e.g. variable constraints in the bottling module). However, finding the optimal solution in one silo of the operation may have negative impact downstream and/or upstream of the whole operation. Consider, for example, the following example. The bottling plant has to process 700,000 litres of Jacob's Creek[14] to satisfy demand. However, the closest (in terms of volume) tank on

[13] The application was developed at SolveIT Software (www.solveitsoftware.com) together with similar applications in other verticals (e.g. for grain handling, pit-to-port mining logistics, and mine-planning activities).

[14] Jacob's Creek is one of the most known, trusted and enjoyed Australian wines around the world.

the tank farm contains 800,000 litres of Jacob's Creek – and what is the optimal decision here? Clearly, from the bottling perspective they should get just 700,000 litres of that wine and process it – but it would be a bad decision from the perspective of another silo: tank farm. They will be left with 100,000 litres leftover with all risks (the quality of these 100,000 litres will go down quickly due to oxygen levels in the tank) and inefficiencies in operations (the tank with the leftover 100,000 litres can't be used in their operation). From tank farm perspective, the 'optimal' decision would be to send 800,000 litres of Jacob's Creek for bottling (no leftovers with clear benefits), but the bottling operation would not be happy with such a solution (note that once a wine is bottled and labelled, the choice for its destination is quite limited as different customers have different requirements for packaging). This is why the global optimum here should consider implications of various decisions across all silos. Further, it is next to impossible to compare the result to some theoretical global optimum – usually the comparisons are made with respect to some metrics from the previous year (i.e. before the system went live). Further, by considering the entire supply chain we can attempt to answer some key (global) questions, as "What is the impact of increasing demand by 10% of Jacob's Creek across all operations"? And the answers for such questions are sought today by businesses and industries.

Darrell Whitley (the former editor-in-chief of Evolutionary Computation Journal who also participated actively in a few real-world projects), wrote:[15] "*Your comments about silos and up-stream and down-stream side effects is spot-on and is something I have worried about for some years now. If you say 'I ran an experiment and I can increase line-loading by 10 percent, what does this do to inventory?' Is that increase in line-loading sustainable or just temporary because if you increase line-loading, you decrease inventory, which might reduce your ability to line-load. When you introduce a change in one silo you move the steady-state in other silos. You almost NEVER see this in academic studies*."

A decision-support system that optimises multi-silo operational problems is of a great importance for an organisation; it supports what-if analysis for operational and strategic decisions and trade-off analysis to handle multi-objective optimisation problems; it is capable of handling and analysing variances; it is easy to modify – constraints, business rules, and various assumptions can be re-configured by a client. Further, from end-user perspective, such decision-support system must be easy to use, with intuitive interfaces which lead to faster and easier adoption by users with less training.

However, it seems to me that the research done within the EC community diverges further and further from complexities of today's problems mainly because the focus of the research is on issues which are secondary for real-world applications. This is partially due to the reasons identified by Jeff Ullman and discussed in the *Inspiration* section of this article – most researchers follow up of some previous work of other researchers, making progress on some 'open issues', and, in the words of Jeff Ullman: "*it almost guarantees that after a while, the work is driven by what **can** be solved, rather than what **needs** to be solved."* Thus the influence of many research papers in the EC community, beyond the world of paper-writing, is minimal.

[15] Private correspondence.

In my recent correspondence with Hans-Georg Beyer he identified: "*... 3 fields where theory should focus on or contribute to in the future: (1) mathematical characterization of evolutionary dynamics (in a sense of a predictive theory); (2) population sizing rules for highly multimodal optimization problems, and (3) development of stopping rules based on evolutionary dynamics*." However, it is not clear to me how even breakthrough results in these areas would help practitioners for approaching complex real-world problems of growing complexity, as described earlier?

Consider, for example, the number of published papers on EAs in dynamic environments. Most researchers focused at recovery rates in cases there was a change in the shape of a landscape. I think it is of no significance, as in real-world problems (1) the objective is usually fixed (e.g. say, you minimize the cost of some operation and you do not change this objective), (2) constraints are changing (e.g. failure of a truck), and (3) we deal with partially-executed solutions. Thus the concept of recovery is very different. There is an enormous gap between theoretical models and practice. The same is in many other research areas. For example, I remember that I was amazed when (over 20 years ago) I discovered the first textbook on genetic algorithms (Goldberg, 1989). The book was around 400 pages, and it included one paragraph on how to deal with constraints (!). The same is true today – many theoretical results are achieved in constraints-free environments and their applicability to real world situations is quite limited.

Further, as discussed in Section 2, it is unclear what is (and what is not) an Evolutionary Algorithm. On one hand, EA practitioners usually employ hybrid forms of evolutionary algorithms (e.g. extending the system by problem-specific initialisation, problem-specific operators) and a successful application of EA to a complex business problem requires a significant dose of 'art'; on the other hand most of theoretical research concentrates on classic versions of EAs and toy problems. There are many research papers published in the EC community on convergence properties of evolutionary algorithms, diversity, exploration, exploitation, constraint-handling, multi-objective optimisation, parallel EAs, handling noise and robustness, ruggedness of the landscape, deceptiveness, epistasis, pleiotropy – to name just a few areas of research. In most cases some standard versions of EA are studied: binary coded GA with a tournament selection or (1+1) ES. Whatever the results, their applicability to solving complex real-world problem are questionable.

A large portion of the research on Evolutionary Algorithms is experimental – hence the researchers use a variety of benchmark functions and test cases. However, these experiments are usually conducted on simple silo problems. The researchers use some classic sets of functions (from f_1,..., f_5 proposed by Ken De Jong [De Jong,1975] to f_n today, where n approaches 100) for numerical optimisation and classic benchmarks (e.g. on graph colouring, traveling salesman, vehicle routing, job shop scheduling) for combinatorial optimisation. However, these small-scale silo problems are far cry from complexity of real-world problems – consequently these theoretical and experimental results are of little help (if any) to any practitioner who works on EA-based enterprise software applications. There are hundreds (if not thousands) of research papers addressing traveling salesman problems, job shop scheduling problems, transportation problems, inventory problems, stock cutting problems, packing

problems, etc. While most of these problems are NP-hard and clearly deserve research efforts, it is not exactly what real-world community needs. Most companies run complex operations and they need solutions for complex multi-silo problems with all their complexities (e.g. many objectives, noise, constraints). In the same time Evolutionary Computation offers various techniques to experiment with, e.g., cooperative coevolution [Potter and De Jong, 1994], where several EAs, each corresponding to a single silo, are run in parallel. Communication between silos may occur during evaluation of individual solutions. Solutions from one silo might be evaluated based on their performance when combined with representative solutions from the other silos.

However, there are very few research efforts which aim in the direction of optimising interdependent multi-silo operational problems with many conflicting objectives, complex constraints and business rules, variability issues and noise. This might be due to the lack of benchmarks or test cases available. It is also much harder to work with a company on such global level as a delivery of successful software solution usually involves many other (apart from optimisation) skills, from understanding the company's internal processes to complex software engineering issues. And it is much harder to report the results, as they may involve revealing company's confidential data. It is also much harder to run significant number of experiments to satisfy requirements of many journals.

Further, in almost in all cases a new method (whether new representation, new set of operators, new selection method, a novel way to incorporate problem-specific knowledge into the algorithm, a novel way to adapt parameters in the algorithm, and so on) is tested against such accepted benchmarks – and the most popular quality measure of a new method is the closeness of the generated solution to the known, global optima in a number of function evaluations. Of course, it is helpful that some silo problems are used for testing, as for many of these the global optima are known (or it is possible to estimate their values). However, for many real-world applications getting to the global optima is secondary. First, the concept of global optima in business is different to that in academia – a global optimum for a silo is referred to as a local optimum solution, as it does not take into account other interacting silos of the business. And the global optimum solution refers to whole multi-silo operation of the organisation. Second, for large-scale (e.g. multi-silo) problems, it would take days (if not more) to generate a global optimum solution, while decision-makers have minutes to react. Third, the multi-silo environment is highly variable (delays, unexpected orders, failures, etc.) and a robust, quality solutions are of higher importance, as the current solution would be modified, anyway, due to changes in the environment. Fourth, due to many, possibly conflicting, objectives, business rules, and soft constraints, the meaning of the term "global optimum" is not that clear – even experienced decision makers often have difficulties in pointing to a better solution out of two available solutions. Finally, the name of the game in industry is not to find an elusive global optimum, but rather to match (and hopefully improve) the results of the human team of experts[16], who have been involved in particular decision-making

[16] There is a nice, one sentence summary of how evolution works: "You don't have to outrun the bear, but you just have to outrun the other hunter".

activity for a significant time – however, technical journals reject such comparisons. Not to mention that it would be extremely hard to document such comparisons in a systematic way without revealing sensitive data of an organisation.

Each EA possesses a number of algorithm-specific parameters. Clearly, theory should provide guidelines how to choose those parameters (this is what Dave Davis referred to in his talk from 15 years ago). But theory can only consider simple toy problems. Many researchers would like to believe that if these toy problems cover certain aspects of real-world problem, the results of the theory can be used as a first guideline to choose these parameters. But it seems to me that this is just a wishful thinking – there is no comparison in terms of complexity between real-world problems are and toy problems – and I cannot see any justified transition of results. Further, most research papers focus on one selected aspect of a problem, whether this is constraint-handling method, handling many objectives, dealing with noise and/or uncertain information. In business problems, however, all these aspects are usually present in every problem – and there is hardly a paper which addresses problems of such complexity. Further, real-world applications usually require hybrid approaches – where an 'evolutionary algorithm' is loaded with non-standard features (e.g. decoders, problem-specific variation operators, memory) – but the current theory of evolutionary algorithms does not support such hybrid approaches very well.

5 Conclusions and Recommendations

Let us conclude by returning to the questions from the second paragraph of the abstract: What are the practical contributions coming from the theory of Evolutionary Algorithms? Did we manage to close the gap between the theory and practice? How Evolutionary Algorithms do compare with Operation Research methods in real-world applications? Why do so few papers on Evolutionary Algorithms describe the real-world applications? For what type of problems Evolutionary Algorithm is "the best" method? Let's address these questions in turn.

5.1 What Are the Practical Contributions Coming from the Theory of Evolutionary Algorithms?

It seems that the practical contributions coming from the theory of Evolutionary Algorithms are minimal at the best. When a practitioner faces complex business problem (problem which involved many silos of the operation, significant number of business rules and constraints, many (possibly conflicting) objectives, uncertainties and noise – all of these combined together, there are very few hints available which might be used in the algorithms being developed. None of the results on convergence properties, diversity, exploration, exploitation, ruggedness of the landscape, deceptiveness, epistasis, pleiotropy that I know of would help me directly in developing EA-based enterprise software application. This is partially due to the lack of benchmarks or test cases of appropriate complexity and partially that the EC technical journals are not appropriate for publicizing business successes (see later questions/answers).

5.2 Did We Manage to Close the Gap between the Theory and Practice?

No, we did not – and as a matter of fact, the gap is growing. As discussed in this article, there are two main reasons for this phenomenon (1) Growing complexity of real-world problems and (2) focus of research community on issues which are secondary for real-world applications.

5.3 How Do Evolutionary Algorithms Compare with Operation Research Methods in Real-World Applications?

It seems to me that (after talking to many practitioners at many major corporations) they compare quite poorly… The main reason (I think) is due to the wide spread of standard Operation Research methods which dominated optimisation aspects of business operations for more than 30 years. Further, the Operation Research community has a few standard and powerful tools (e.g. integer programming methods) which are widely in use in many organisations. On the other hand, this is not the case for the EC community – there are no 'plug-in' software tools appropriate to deal with thousands of variables and hundreds of constraints, there are no tools available of comparable power to integer programming. Further, many Operation Research methods are exact – they guarantee the optimum solution, which is not the case with heuristic methods in general and Evolutionary Algorithms in particular.

However, there is one catch here, as every time we solve a problem we must realize that we are in reality only finding the solution to a *model* of the problem. All models are a simplification of the real world – otherwise they would be as complex and unwieldy as the natural setting itself. Thus the process of problem solving consists of two separate general steps: (1) creating a model of the problem, and (2) using that model to generate a solution:

$$\text{Problem} \rightarrow \text{Model} \rightarrow \text{Solution}$$

Note that the "solution" is only a solution in terms of the model. If our model has a high degree of fidelity, we can have more confidence that our solution will be meaningful. In contrast, if the model has too many unfulfilled assumptions and rough approximations, the solution may be meaningless, or worse.

So in solving real-world problem there are at least two ways to proceed:

1. We can try to simplify the model so that traditional OR-based methods might return better answers.
2. We can keep the model with all its complexities, and use non-traditional approaches, to find a near-optimum solution.

In either case it's difficult to obtain a precise solution to a problem because we either have to approximate a model or approximate the solution. In other words, neither exact methods nor heuristic methods return optimum solution to *the problem*, as the former methods simplify the problem (by building simplified, usually linear, model of the problem) so the optimum solution to the simplified model does not correspond to the optimum solution of the problem, and the latter methods return near-optimum

solutions (but to the more precise models). Further, as discussed in section 4, for many real-world applications the issue of getting to the global optima is secondary as robust, quality (i.e. near-optimum) solutions are of higher importance. And the more complexity in the problem (e.g., size of the search space, conflicting objectives, noise, constraints), the more appropriate it is to use a heuristic method. A large volume of experimental evidence shows that this latter approach can often be used to practical advantage.

5.4 Why Do So Few Papers on Evolutionary Algorithms Describe Real-World Applications?

The journal editors and reviewers of submitted papers are well versed in the standard criteria for acceptance: (a) clearly revealed algorithms, so the reader can at least attempt to replicate you approach, (b) well characterized problems, so the reader can tell if his problem is the right type, (c) rigorous comparison with known results, so the reader can have confidence your results are significant. All this is needed for verifiability – the soul of science. On the other hand, a successful application of EA to complex business problem requires a significant dose of 'art' – and technical journals and (to some lesser extend) conferences generally have difficulties with that. For additional thoughts on this very topic, see (Michalewicz, 2012).

5.5 For What Type of Problems Evolutionary Algorithm Is "The Best" Method?

A submission of a survey article on Evolutionary Algorithms (written with Marc Schoenauer) – for Wiley Encyclopedia of Operations Research and Management Science, 2010 edition – generated editor's comment/request: "*Can the authors provide objective guidance on the types of problems for which evolutionary methods are more appropriate than standard methods? I know a lot of 'hearsay' related to this issue but I am wondering if there is more objective evidence. Is there any solid evidence of EA superiority in a class of problems*?" More and more people question the usefulness and applicability of Evolutionary Computation methods and it is essential that our community would get ready to answer such questions.

And I think that the right answers for above questions are not of the type: "EA techniques are superior for, say, symmetrical TSP" or "EA techniques are superior for, say, such-and-such types of scheduling problems," as the main issue is just in *size* and *complexity* of the problems – for example, multi-silo operational problems with many conflicting objectives, tens of thousands of variables, complex constraints and business rules, variability issues and noise, interdependencies between operational silos – problems, for which standard Operations Research methods are not appropriate. For such problems the business starts looking for optimization support because rational decision making and logical decomposition of the problem are no longer possible. This is a big chance for Evolutionary Computation community, and the time is right to move that direction. However, this would require some fundamental changes in a way the EC community looks at real-world problems…

Many traditional decision-support applications at large corporations worldwide have failed, not realising the promised business value, mainly because small improvements and upgrades of systems created in the 1990s do not suffice any longer for solving 21st century companies' problems. Also, the existing applications often are not flexible enough to cope with exceptions, i.e. it is very difficult, if not impossible, to include problem-specific features – and most businesses have unique features which need to be included in the underlying model, and are not adequately captured by off-the-shelf standard applications. Thus, the results are often not realistic. A new approach is necessary which seamlessly integrates local models and optimisation algorithms for different components of complex business problems with global models and optimisation algorithms for the overall problem. Such decision-support systems should allow also manual adjustments by domain experts, to achieve optimal decisions with the flexibility to be adapted to business rules and unforeseen circumstances. And I believe that this new generation of decision-support systems will be based on Evolutionary Algorithms (understood in a broad sense of this term).

So let's move to the last part of this article and discuss what should be done to remedy the current state of the art with respect to Evolutionary Algorithms? Before we start this discussion, let's go back 20 years…

Most of the readers might be aware that the field of evolvable hardware (EHW) is about 20 years old. Some of the first really serious work was done by John Koza – in fact, EHW was one of the applications Koza used to show the power of Genetic Programming (GP). He used GP to evolve a series of analog filters (both circuit configuration and passive component values). He was able to evolve a series of filters that were essentially identical to filters that were patented in the 1930s for the telephone industry. Based on those results he coined the phrase "human competitive designs" and claimed this was something GP can do. Ten years ago there was the annual (small, but enthusiastic) NASA/DOD sponsored EHW conference. However, that conference is no longer held and a majority of the main players have moved on to other areas. Why? Well, the human competitive designs were just that and little else. GP could duplicate what people had done before, but really didn't (couldn't?) evolve much new and innovative. Consequently people figured all of the low hanging fruit had been picked and EHW had little new to offer. So they moved on to new areas…

This story should serve as a warning to the whole EC community – I believe that the future of EAs would be determined by the applicability of EAs. The excitement connected with terms like 'genetic', 'evolution', 'emergence', if not supported by practice, would wear off (this is, to some extent, already happening in industry). And as long as EAs would produce just 'interesting', 'promising', or 'comparable to an OR method' results on some benchmark problems, and the theory would focus on standard properties of EAs, it would be hard to compete in the real-world environment.

Some theoreticians do not worry: one of them wrote to me: "*Good problem solvers/theoreticians can also work in other fields. I do not see this as a severe problem.*" I found such comments quite depressing, as they remind me of a sad story where a group of scientists (astronomers) were studying climate patterns of a planet X which

was located millions of light-years from the Earth. One day the planet exploded, and these scientists moved to study another planet, Y…

So, what should we do? What can be done? Well, apart from suggesting that the IEEE should give a post mortem Evolutionary Computation Pioneer Award to Isaac Newton, it seems there are a few things that are worth considering; these include:

1. revisiting the policies of some EC journals – and introduce a different set of criteria to evaluate application papers. After all, the whole purpose of the journal is to disseminate knowledge. It assumes that only the top researchers, the leading edge theory types, are the only ones who might read the journals. Garry Greenwood summarized it nicely: "*The notion that a paper has to have some theoretical component or it doesn't rise to the level of a journal publication is absurd*". However, due to complexities of evaluating the merit of an application paper, this task is far from trivial. There is need for a framework that would give the reader confidence that: (a) the described methods really are an advance on previous industrial practice (validity); (b) the described methods will often port to other industrial problems (generalisability), and (c) the advances described really are new (novelty). And there are no easy ways to accomplish that. Note that application papers that describe systems which are in daily use (whatever their significance) are routinely rejected as, for example, it is impossible to run millions scenarios, which is usually required for the evaluation of the approach. However, papers which document millions of scenarios run on a meaningless problem (e.g. an unconstrained sphere problem) have much better chance – as this is real science… Whichever way I look at this, it does not seem right.
2. removing 'lip-service' from publications, e.g. the authors of accepted papers should be asked to remove sentences about applicability of EAs to a wide-range of real-world problems in many verticals (as discussed in section 3 of this paper), not to misled the EC community (unless some concrete examples are provided).
3. educating the research community on real-world problems, as it seems that most of the researchers are disconnected from the real-world and its challenges. Some of these challenges include separation of business rules and constraints from the optimisation engine, as no industrial user is going to accept a software that needs to be modified (or worse, redesigned) if the application problem slightly changes. The methods have to be robust in the sense they can work with just about any variation of the application problem domain without redesign or recoding – but rather just by changing environmental settings in the configuration file of the system[17]. Garry Greenwood, summarised this[18] very well: "*If you have to take your EC method offline to change it for every new application problem variation and the user interface keeps changing,*

[17] Similar issue was raised in [Goertzel, 1997]: AI-based software applications are quite fragile, often a slight change in problem definition will render them useless. This problem is the main motivation for the establishment of Artificial General Intelligence community.

[18] Private correspondence.

forget it." Additional challenges include "explanatory features" of the optimiser – as usually end-users hate "black-boxes" which return just a recommended result without any "explanations" and interactive components, which are very helpful in influencing the optimiser into particular directions, to name a few.

4. encouraging researchers who deal with theoretical aspects of EAs to pay attention to commercial developments. Theoreticians should understand that the work in vacuum is not that significant. For many people it is clear that a researcher working on internet search engines should not ignore developments at Google – so the same should apply to an EA researcher who works on optimisation of supply chain type of problems and such researcher should be knowledgeable on offerings of SAP (Kallrath and Maindl, 2006).
5. developing artificial problem sets that better reflect real-world difficulties which the research community can use to experience (and appreciate) for themselves what it really means to tackle a real-world problem. This would lead to some meaningful classification of different types of problems to understand the effectiveness of various algorithms. This would require studying the problem representation and modeling issues, as these are the key components in approaching any real-world problem. Finally, more emphasis should be placed on studying the reliability of algorithms versus the frequency of hitting the global optimum, as in the real world setting reliability (in the sense of getting quality solution every run) is more desirable than getting global optimum in 95% of runs (and poor quality solutions in the remaining 5% of runs).

I think the following story nicely concludes this chapter and illustrates the gap between the theory and practice in Evolutionary Algorithms:

A scientist discovered a special dog food with amazing characteristics. He has proved (by a scientific method) that if a dog eats this food on regular basis, its fur would be always shiny, its teeth would be always white, it would never be sick, it would be well-behaved, together with many additional advantages. However, there was just one problem with this invention – when it was commercialized, it was discovered that dogs refused to eat this food…

Acknowledgements. The author is grateful to Manuel Blanco Abello, Brad Alexander, Hans-Georg Beyer, Yuri Bykov, Ken De Jong, Fred Glover, Garry Greenwood, Dennis Hooijmaijers, Pablo Moscato, James Whitacre, and Darrell Whitley for their comments – many of them were incorporated in the final version of this chapter. Also, a few quotes from our private correspondence were included in the text (with permission of the author).

References

[Ackoff, 1979] Ackoff, R.: The Future of OR is Past. JORS (1979)

[Chow & Yuen, 2011] Chow, C.K., Yuen, S.Y.: An Evolutionary Algorithm That Makes Decision Based on the Entire Previous Search History. IEEE Transactions on Evolutionary Computation 15(6), 741–769 (2011)

[De Jong, 1975] De Jong, K.A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral Dissertation, University of Michigan, Ann Arbor, MI. Dissertation Abstract International 36(10), 5140B (University Microfilms No 76-9381) (1975)

[De Jong, 2002] De Jong, K.A.: Evolutionary Computation: A unified approach. Bradford Book (2002)

[Gattorna, 2010] Gattorna, J.: Dynamic Supply Chains. Prentice Hall (2010)

[Goertzel, 1997] Goertzel, B.: From Complexity to Creativity: Explorations in Evolutionary, Autopoietic, and Cognitive Dynamics. Plenum Press (1997)

[Goldberg, 1989] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)

[Hinterding et al. 1999] Hinterding, R., Michalewicz, Z.: Your Brains and My Beauty: Parent Matching for Constrained Optimisation. In: Proceedings of the 5th IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, pp. 810–815 (1998)

[Ibrahimov et al. 2011] Ibrahimov, M., Mohais, A., Schellenberg, S., Michalewicz, Z.: Advanced Planning in Vertically Integrated Supply Chains. In: Bouvry, P., González-Vélez, H., Kołodziej, J. (eds.) Intelligent Decision Systems in Large-Scale Distributed Environments. SCI, vol. 362, pp. 125–148. Springer, Heidelberg (2011)

[Kallrath and Maindl, 2006] Kallrath, J., Maindl, T.I.: Real Optimization with SAP-APO. Springer (2006)

[Michalewicz, 1992] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 1st edn. Springer (1992)

[Michalewicz, 2012] Michalewicz, Z.: The Emperor is Naked: Evolutionary Algorithms for Real-World Applications. ACM Ubiquity (2012)

[Mohais et al. 2011] Mohais, A., Ibrahimov, M., Schellenberg, S., Wagner, N., Michale wicz, Z.: An Integrated Evolutionary Approach to Time-Varying Constraints in Real-World Problems. In: Chiong, R., Weise, T., Michalewicz, Z. (eds.) Variants of Evolutionary Algorithms for Real-World Applications. Springer (2011)

[Potter and De Jong, 1994] Potter, M.A., De Jong, K.A.: A Cooperative Coevolutionary Appro ach to Function Optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994)

[Ullman, 2009] Ullman, J.D.: Advising Students for Success. Communications of the ACM 53(3), 34–37 (2009)