Engineering Letters, 13:2, EL_13_2_11 (Advance online publication: 4 August 2006)

Hybrid Genetic Algorithms: A Review

Tarek A. El-Mihoub, Adrian A. Hopgood, Lars Nolle, Alan Battersby

Abstract—Hybrid genetic algorithms have received significant interest in recent years and are being increasingly used to solve real-world problems. A genetic algorithm is able to incorporate other techniques within its framework to produce a hybrid that reaps the best from the combination.

In this paper, different forms of integration between genetic algorithms and other search and optimization techniques are reviewed. This paper also aims to examine several issues that need to be taken into consideration when designing a hybrid genetic algorithm that uses another search method as a local search tool. These issues include the different approaches for employing local search information and various mechanisms for achieving a balance between a global genetic algorithm and a local search method.

***Index Terms*—Genetic algorithms, evolutionary computation, hybrid genetic algorithms, genetic-local hybrid algorithms, memetic algorithms, Lamarckian search, Baldwinian search.**

I. Introduction

A genetic algorithm is a population-based search and optimization method that mimics the process of natural evolution. The two main concepts of natural evolution, which are natural selection and genetic dynamics, inspired the development of this method. The basic principles of this technique were first laid down by Holland [1] and are well described, for example, in [2],[3].

The performance of a genetic algorithm, like any global optimization algorithm, depends on the mechanism for balancing the two conflicting objectives, which are exploiting the best solutions found so far and at the same time exploring the search space for promising solutions. The power of genetic algorithms comes from their ability to combine both exploration and exploitation in an optimal way [1]. However, although this optimal utilization may be theoretically true for a genetic algorithm, there are problems in practice. These arise because Holland assumed that the population size is infinite, that the fitness function accurately reflects the suitability of a solution, and that the interactions between genes are very small [4].

In practice, the population size is finite, which influences the sampling ability of a genetic algorithm and as a result affects its performance. Incorporating a local search method within a genetic algorithm can help to overcome most of the obstacles that arise as a result of finite population sizes.

Incorporating a local search method can introduce new genes which can help to combat the genetic drift problem [5], [6] caused by the accumulation of stochastic errors due to finite populations. It can also accelerate the search towards the global optimum [7] which in turn can guarantee that the convergence rate is large enough to obstruct any genetic drift.

The Parallel Recombinative Simulated Annealing (PRSA) algorithm [8] fights the genetic drift problem in another way by combining the concept of the cooling schedule of simulated annealing [9], Boltzmann tournament selection [10], and standard genetic operators.

Due to its limited population size, a genetic algorithm may also sample bad representatives of good search regions and good representatives of bad regions. A local search method can ensure fair representation of the different search areas by sampling their local optima [11] which in turn can reduce the possibility of premature convergence.

In addition, a finite population can cause a genetic algorithm to produce solutions of low quality compared with the quality of solution that can be produced using local search methods. The difficulty of finding the best solution in the best found region accounts for the genetic algorithm operator's inability to make small moves in the neighborhood of current solutions [12]. Utilizing a local search method within a genetic algorithm can improve the exploiting ability of the search algorithm without limiting its exploring ability [7]. If the right balance between global exploration and local exploitation capabilities can be achieved, the algorithm can easily produce solutions with high accuracy [13].

Although genetic algorithms can rapidly locate the region in which the global optimum exists, they take a relatively long time to locate the exact local optimum in the region of convergence [14], [15]. A combination of a genetic algorithm and a local search method can speed up the search to locate the exact global optimum. In such a hybrid, applying a local search to the solutions that are guided by a genetic algorithm to the most promising region can accelerate convergence to the global optimum. The time needed to reach the global optimum can be further reduced if local search methods and local knowledge are used to accelerate locating the most promising search region in addition to locating the global optimum starting within its basin of attraction.

The improper choice of control parameters is another source of the limitation of genetic algorithms in solving real-world problems [16] due to its detrimental influence on the trade-off between exploitation and exploration. Depending on these parameters the algorithm can either succeed in finding a near-

Manuscript received February 3, 2006.

T. A. El-Mihoub, A. A. Hopgood, Lars Nolle, and A. Battersby are with School of Computing & Informatics, Nottingham Trent University. Nottingham, NG11 8NS, UK (phone: +44 (0)870 127 8429; fax: +44 (0)115 848 8365; e-mail: tarek.elmihoub, adrian.hopgood, lars.nolle, and alan. Battersby @ntu.ac.uk).

optimum solution in an efficient way or fail. Choosing the correct parameter values is a time-consuming task. In addition, the use of rigid, constant control parameters is in contradiction to the evolutionary spirit of genetic algorithms [17]. For this reason, other search techniques can be utilized to set the values of these parameters whilst the search is progressing.

In this paper, hybrid genetic algorithms are reviewed through presenting the different ways in which the roles of a search method and a genetic algorithm can be integrated. The aim of this presentation is not to classify hybrid genetic algorithms, but to shed light on the possible ways of combining a search method within the framework of a genetic algorithm. However, the reader can refer to [18] for an architectural taxonomy of combinatorial memetic algorithms (MA) [19] and to [20], where meta-heuristics are classified based on the design space and implantation space aspects.

This paper also aims to gain an insight into some of the design issues of hybrid genetic algorithms through reviewing the different mechanisms of utilizing local search information within genetic search and the various techniques to achieve a balance between exploration and exploitation.

II. A Complementary View

Hybrid genetic algorithms, as any hybrid system, are based on the complementary view of search methods [21 p.223]. Genetic and other search methods can be seen as complementary tools that can be brought together to achieve an optimization goal. In these hybrids, a genetic algorithm incorporates one or more methods to improve the performance of the genetic search. There are several ways in which a search or optimization technique can complement the genetic search.

A. Capability Enhancement

A technique can be utilized within a genetic algorithm to enhance search capabilities. A genetic algorithm is normally viewed as a global search method that can capture the global view of a problem domain. Different techniques can be incorporated within a genetic algorithm to improve its performance in different ways. When a genetic algorithm as a global search method is combined with a problem-specific method as a local method, the overall search capability can be enhanced. The enhancement can be in terms of solution quality and/or efficiency. This performance can also be improved by ensuring production of feasible solutions in the case of highly constrained problems. This paper focuses on the global local complementary view of genetic hybrids which have been variously referred to as memetic algorithms (MA) [19], genetic-local search methods [22], Lamarckian genetic algorithms [23], Lamarckian search, and Baldwinian search [24].

Function approximation techniques can also be incorporated in a genetic search to speed up the search. It is also possible to utilize other techniques to replace one or more of the genetic operators in order to overcome some of the problems that face genetic search.

1) Improving Solution Quality

Local search methods and genetic algorithms are usually viewed as two complementary tools. A local search algorithm's ability to locate local optima with high accuracy complements the ability of genetic algorithms to capture a global view of the search space. Holland [1], cited in [25], suggested that the genetic algorithm should be used as a pre-processor for performing the initial search, before invoking a local search method to optimize the final population. Bilchev and Parmee [26], for example, used their ant colony optimization [27] model for continuous search spaces as local search method to improve the quality of the solutions produced by a genetic algorithm in order to solve a real-world, heavily constrained, engineering design problem.

Performing local search on a genetic algorithm's population can introduce diversity and help to resist the genetic drift. It enables fair representation of different search areas in order to fight premature convergence. Incorporating a local search algorithm also introduces an explicit refinement operator which can produce high quality solutions.

2) Improving Efficiency

The efficiency of a local search in reaching a local optimum integrates the efficiency of a genetic algorithm in isolating the most promising basins of the search space. Therefore, incorporating a local search into a genetic algorithm can result in an efficient algorithm. The efficiency of the search can be enhanced in terms of the time needed to reach the global solution, and/or the memory needed to process the population.

a) Convergence Speed

A major concern in genetic algorithm design is efficiency in terms of the time needed to reach a solution of desired quality. In real-world problems, function evaluations are the most time-consuming part of the algorithm. For example, the designers of today's complex engineering systems usually rely on expensive computer analysis and simulation programs, where the execution time for a single function evaluation can be of the order of hours or days [28]. Finite element analysis (FEA), computational fluid dynamics (CFD), heat transfer and vehicle dynamic simulations are examples of such programs. Hybridization in addition to parallelization [29], time utilization [30], and evaluation relaxation (function approximation) can be used to speed up a genetic search [31].

Genetic algorithms often show significant improvements in search speed when combined with local search methods utilizing domain-specific knowledge [20], [32]. There is an opportunity in hybrid optimization to capture the best of both schemes [13]. This is the reason why genetic hybrids are being increasingly used to solve real-world problems. Different search methods have been mixed with genetic algorithms in real-world applications [15], [22], [33-37].

b) Population Size

Population size is crucial in a genetic algorithm. It determines the memory size and the convergence speed in serial genetic algorithms and affects the speed of search in the case of parallel genetic algorithms. Efficient population sizing

is critical for getting the most out of a fixed budget of function evaluations. The gambler's ruin model [38] was used to estimate the population size of genetic algorithms. This model was used to show that population size depends on two parameters, which can be affected by incorporating local search. The two parameters represent the standard deviation of the population and the signal difference between the best and second best building blocks. If a local search method is incorporated in such a way as to reduce the standard deviation of the population and to increase the signal difference between the best and the second best chromosome, the resulting hybrid can be efficient even with small population sizes. Espinoza et al. [39] showed the effect of a local search method on reducing the population size, compared to a pure genetic algorithm. El-Mihoub et al. [40] demonstrated the combined effect of probability of local search and learning strategy on the population size requirements of a hybrid.

3) Guarantee Feasible Solutions

In highly constrained optimization problems, the crossover and mutation operators generally produce illegal or infeasible solutions and hence waste search time. This problem can be solved by incorporating problem-specific knowledge. Problem-specific knowledge can be used either to prevent the genetic operators from producing infeasible solutions or to repair them.

The partial matched crossover (PMX) [41] was proposed for use in order-based problems to avoid the generation of infeasible solutions. Grefenstette et al. [42] suggested a heuristic crossover operator that could perform a degree of local search for the traveling salesman problem (TSP). Davidor [43] designed "analogous crossover" where local information is used to decide which crossover sites can produce unfit solutions. Heuristic crossover operators were used to solve a timetabling problem in order to ensure that the most fundamental constraints are never violated [44]. Freisleben and Merz [45] proposed the distance preserving crossover (DPX) to produce feasible solutions to solve TSP without losing diversity. They used the non- sequential 4-change [46] as a mutation operator for the same reason. Cycle crossover (CX) [47], order crossover (OX) [47], matrix crossover (MX) [48], modified order crossover (MOX) [49], edge recombination crossover (ERX) [50], 2-opt operator [51], 3-opt operator [51] and or-opt operators [51] are examples of crossover and mutation operators which have been developed for TSP. A special edge recombination crossover [52] has been constructed for the three-matching problem (3MP). The crossover operator has been replaced with the gene-pooling operator to produce feasible solutions when optimizing the number and positions of fuzzy prototypes for efficient data clustering [53].

A problem-specific knowledge search method can be used to recover the feasibility of solutions generated by the standard genetic operators. Repairing such solutions can help the genetic search to avoid the danger of premature convergence, which occurs when all or most solutions are infeasible [54], [55]. The force feasible heuristic operator [56] was used to solve the problem of scheduling aircraft landing times. Konak and Smith [57] combined a genetic algorithm with a cut-saturation algorithm for the backbone design of communication networks. They use a uniform crossover operator with a K-node-connectivity repair algorithm to repair infeasible offspring. Areibi and Yang [58] used repair heuristics in their proposed approach to solve VLSI circuit layout. The approach combines a hierarchical design technique, genetic algorithms, constructive techniques, and advanced local search. They also used the OX operator to avoid infeasible solutions in solving VLSI design problems.

4) Fitness Function Estimation

If the fitness function is excessively slow or complex to evaluate, approximation function evaluation techniques can be utilized to accelerate the search without disrupting search effectiveness. This is because genetic algorithms are robust enough to achieve convergence in the face of noise produced by the approximation process. Fitness approximation schemes replace high-cost accurate fitness evaluation with a low-cost approximate fitness assignment procedure. This can be achieved either by evolutionary approximation, where the fitness of a chromosome is estimated from its parents' fitness, or function approximation, where the fitness function is replaced by an alternate simpler model. Jin [59] provides a comprehensive survey on fitness approximation techniques.

The selection of an appropriate approximation model to replace the real function is an important step in ensuring that the optimization problem is solved efficiently. Neural network [21 ch. 8] models have widely been used for function approximation [60]. Willmes et al. [61] compared neural networks and the Kriging method for constructing fitness approximation models in evolutionary algorithms. Jin and Sendhoff [62] combined the k-nearest-neighbor clustering method and a neural network ensemble to estimate a solutions' fitness. Burdsall and Giraud-Carrier [53] used an approximation of the network's execution to evaluate solutions fitness instead of constructing a radial basis function network (RBF) to optimize the topology of a neural network. The approximation is based on an extension of the nearest-neighbor classification algorithm to fuzzy prototypes. Ankenbrandt et al. [63] implemented a system of fuzzy fitness functions, to grade the quality of chromosomes, representing a semantic net. The system is used to assist in recognizing oceanic features from partially processed satellite images. Pearce and Cowley [64] presented a study of the use of fuzzy systems to characterize engineering judgment and its use with genetic algorithms. They demonstrated an industrial design application where a system of problem-specific engineering heuristics and hard requirements are combined to form a fitness function.

5) Operation Substitution

Genetic algorithms present a methodological framework that is easy to understand and handle. This framework is open to the incorporation of other techniques [65]. It is possible to utilize other techniques to perform one or more of the genetic algorithm operations. These incorporated techniques can be used to replace either the crossover operator, mutation operator or both.

In probabilistic model-building genetic algorithms (PMBGA) or estimation of distribution algorithms (EDA) [66], a probabilistic model is utilized to learn the structure of a problem on the fly. This model is used instead of the standard genetic operators to ensure a proper mixing and growth of building blocks. These algorithms replace the standard crossover and mutation operators of genetic algorithms, by building a probabilistic model that estimates the true distribution of promising solutions. New potential solutions are then generated by sampling this model. Population based incremental learning (PBIL) [67], univariate marginal distribution algorithm (UMDA), compact genetic algorithm (CGA), bivariate marginal distribution algorithms (BMDA), factorized distribution algorithms (FDA) and the Bayesian optimisation algorithm (BOA) [68] are all examples of PMBGA that are reported to have a better search ability, than that of the simple genetic algorithm, in solving a broad class of problems [66]. Tsutsui et al. [69] proposed the aggregation pheromone system (APS), which introduced the concept of pheromone trail of the ant colony optimization [27] into the PMBGAs, to solve real-valued optimization problems.

Leng [70] proposed the guided genetic algorithm (GGA) which is a hybrid genetic system that borrows the concept of feature and penalties from the guided local search (GLS) [71]. The GGA modifies the fitness function by means of penalties to escape local optima. Two specialized crossover and mutation operators, which are biased by the penalties to change genes that are involved in more penalties, are used in order to explore the search space.

When a problem-specific representation is used in a genetic algorithm, the standard genetic variation operators are usually replaced with problem-specific operators. Hedar and Fukushima [72] replaced the ordinary crossover with a simplex crossover that produces a simplex offspring from mating simplex parents (is the dimension of the problem to be solved). In this hybrid, a mutation operator, which is more suitable for simplex representation, was used. Quantum-inspired genetic algorithms [73]-[75] borrow the concepts of quantum-bits and -states superposition from quantum computing. In these algorithms, the individuals are represented as a string of quantum-bits. Quantum-gates are then used to modify these individuals instead of crossover and mutation operators. The power of these algorithms comes from the great diversity they provide by using quantum coding. Each single quantum individual in reality represents multiple classical individuals. The results reported from using this hybridization to solve combinatorial and continuous optimization problems are promising.

Tan et al. [76] replaced the standard mutation operator by simulated annealing [9] to solve system identification and linearization problems. The results showed a more accurate search and faster convergence when compared with a pure genetic algorithm. The multi-step crossover (MSX) [77] was proposed to solve combinatorial optimization problems. Riopka and Bock proposed a collective learning genetic algorithm [78], in which an intelligent recombination based on the exchange of knowledge between chromosomes, is used to effectively find high quality solutions to combinatorial optimization problems. Magyar et al. [52] introduce several heuristic crossover and local hill-climbing operators to solve the three-matching problem. Fundamental to the technique here is the adaptation of the selected operator. Two fuzzy connective-base (FCB) crossover operators types (dynamic and heuristic) have been proposed in [79] for real-coded genetic algorithms to fight premature convergence problems.

B. Optimizing the Control Parameters

The setting of genetic algorithm control parameters is a key factor in the determination of the exploitation versus exploitation trade-off. Other techniques can be used to monitor the behavior of a genetic algorithm in order to adapt its control parameters to improve the search performance. The ability of fuzzy logic to represent knowledge in imprecise and non-specific ways enables it to be used to reason on knowledge that is not clearly defined or completely understood. This ability makes fuzzy logic a suitable choice for adapting the control parameters of a genetic algorithm. Fuzzy logic has allowed a small group of researchers to devise ways of optimizing performance and solution quality of genetic algorithms [80]. It is used to incorporate the many heuristics and techniques of experienced genetic algorithm researchers into fuzzy logic systems in order to adapt the control parameters. The goal of such a system is generally to avoid undesirable behaviors such as premature convergence and to speed up the convergence of the genetic algorithm [81].

It is also possible to incorporate a genetic algorithm within another technique to optimize control parameters, since genetic algorithms are in practice very effective optimization techniques. A genetic algorithm can be applied to the optimization of a neural network in a variety of ways. It can be utilized to adjust the neural network weights [82]-[84] their topology [85]-[88] and learning rules [89], [90]. For a comprehensive review of evolving neural networks the reader can refer to [91]. Karr [92] described an application to the cart-pole balancing system and used a genetic algorithm to evolve the membership functions of a fuzzy controller. The resulting, optimized fuzzy logic controller performed better than the controller based on membership functions designed by a human expert. These promising results have been confirmed by an application of the method for online control of a laboratory pH system with drastically changing system characteristics [93]. Genetic algorithms can also be used to automate the learning of fuzzy control rules [94]. They have also been used to optimize the control parameters of ant colony optimization algorithms [95]-[97].

III. Hybrid Design Issues

Incorporating a search method within a genetic algorithm can improve the search performance on the condition that their roles cooperate to achieve the optimization goal. There is an opportunity in hybrid optimization to capture the best of both schemes [13]. This opportunity depends on the design details of the hybrid genetic algorithm. There are several issues that

need to be taken into consideration when designing a hybrid genetic algorithm. Some of the design choices faced by hybrid practitioners while solving real-world problems are discussed here.

Due to their major impact on hybrid genetic performance, the discussion is concentrated on the strategies of utilizing local search information within a hybrid, and mechanisms that can be used to achieve a balance between exploration and exploitation. First, the relation between local search and learning, and its different models, are presented. Then, different techniques that can be used to achieve the optimal division of labor between the global genetic algorithm and the local search method are reviewed.

A. Local Search and Learning

Local search methods use local knowledge to improve a solution's chances to propagate its characteristics into the next generations. Due to the similarities in the role of the local search within the genetic search and the role of learning within the evolution process, the local search is usually viewed as a learning process.

The way by which gained information through local search is utilized within a hybrid genetic algorithm has a great impact on the performance of the search process. Two basic approaches based on biological learning models have been adopted to utilize local information; the Lamarckian approach and the Baldwinian approach [98]. There is also a third model, which is a mixture of the basic models and its effectiveness has been proven in solving real-world problems [55], [99]-[101].

1) Lamarckian Learning

The Lamarckian approach is based on the inheritance of acquired characteristics obtained through learning. This approach forces the genetic structure to reflect the result of the local search. The genetic structure of an individual and its fitness are changed to match the solution found by a local search method. In the Lamarckian approach, the local search method is used as a refinement genetic operator that modifies the genetic structure of an individual and places it back in the genetic population.

Lamarckian evolution, in spite of being recognized as never occurring in biological systems due to the lack of a mechanism to accomplish it, can be simulated in a computer in order to shed light on issues of general evolvability. Lamarckian evolution can accelerate the search process of genetic algorithms [102]. On the other hand, by changing the genetic structure of individuals, it can disrupt schema processing which can badly affect the exploring abilities of genetic algorithms. This may lead to premature convergence [102]. When a Lamarckian approach is adopted, inverse mapping from phenotype to genotype is required. The inverse mapping may be computable in many simple applications. However, for real-world problem solving, the computation will typically be intractable [103]. Most of hybrid genetic algorithms that repair chromosomes to satisfy constrains are Lamarckian and the technique has been particularly effective in solving TSP [24].

2) Baldwinian Learning

The Baldwin learning allows an individual's fitness to be improved by applying a local search, whereas the genotype remains unchanged. In this way, it improves the solution's chances to propagate its structure to the next generations. Like natural evolution, learning does not change an individual's genetic structure, however it increases its chances of survival. The Baldwinian approach, in contrast to the Lamarckian one, does not allow parents to pass their learned or acquired characteristics to their offspring. Instead, only the fitness after learning is retained. A local search method in the Baldwinian approach is usually used as a part of the individual's evaluation process. The local search method uses local knowledge to produce a new fitness score that can be used by the global genetic algorithm to evaluate the individual's ability to be improved.

The Baldwin effect is somewhat Lamarckian in its results although it uses different mechanisms [103]. It explains interactions between learning and evolution by paying attention to balances between benefit and cost of learning. The Baldwin effect consists of the following two steps [104]. In the first step, learning gives individuals the chance to change their phenotypes to improve their fitness. Individuals, who found learning useful and help their fitness to improve, will spread in the next population. In the second step, if the environment is sufficiently stable, the cost associated with learning results in selection favoring individuals that have the traits, which are acquired by others through learning, already coded into their genotype. Through this mechanism, called genetic assimilation, learning can accelerate the genetic acquisition of learned traits indirectly. A critical precondition for genetic assimilation appears to be a strong correlation between genotype and phenotype space so that nearness in the phenotype space implies nearness in the genotype space [105]. Otherwise, the acquired traits have little chance of eventually becoming encoded in the genome via chance through genetic operations.

Hinton and Nolan [98] illustrated how the Baldwin effect can transform the fitness landscape of a difficult optimization problem into a less difficult one, and how the genetic search is attracted toward the solution found by learning. Gruau and Whitley [11] showed how local search can change the landscape of fitness function into flat landscapes around the basin of attraction. This change in fitness landscape is known as the smoothing effect. They demonstrated the impact of the smoothing effect on the search process. This learning strategy could be more effective but slower than the Lamarckian approach, since it does not disrupt schema processing of genetic algorithms [102]. Baldwinian search can also have the effect of obscuring genetic differences and, thus, hindering the evolution process [105]. This is known as the hindering effect. Essentially, this occurs as a result of different genotypes mapping to the same or similar phenotypes (as a result of the smoothing effect) with equivalent fitness scores being produced. The genotypes cannot be effectively discriminated according to their fitness values without considering the learning cost and the evolution of effective solutions is

hindered. The hindering effect can also obstruct the ability of the Baldwinian search to self-adapt the local-search-duration control parameter [106]. The Baldwinian effect can aggravate the problem of multiple genotype to phenotype mappings [24], [99]. This problem can also waste the resources of hybrids that use clustering techniques in the genotype domain to reduce unnecessary local search, in contrast to the Lamarckian approach which has been shown to help alleviate this problem [107].

Hart et al. [108] pointed to the importance of considering the cost of learning, which has been ignored by most researchers when studying the impact of the Baldwinian strategy on the hybrid search by analyzing its performance based on the number of generations of the genetic algorithm only. Learning can introduce a computational cost which overweighs its benefits in search.

3) Hybrid Lamarckian-Baldwinian Models

Hybrid Lamarckian-Baldwinian models are created with a view towards combing the advantages of both forms of learning models [55]. The combination of the Baldwinian and the Lamarckian approaches can be done at two different levels. Hybridization can be used at the individual-level, where some individuals evolve using the Lamarckian approach while the other individuals evolve using the Baldwinian approach [99], [100]. Houck et al. [99] found that this form of partial Lamarckian approach outperformed both the pure Lamarckian and the pure Baldwinian approaches on a selected set of test problems. The other level is the gene-level, where a number of genes evolve using the Lamarckian strategy and the remaining genes evolve using the Baldwinian approach [101]. This approach was used to solve the sorting network problem. It can reduce the problem search space and help to produce an efficient search [101].

The adoption of any form of learning in a hybrid genetic algorithm has a great impact on its performance. Several researchers have investigated how these different leaning strategies affect the performance of hybrid genetic algorithms by comparing them with pure genetic algorithms. Gruau and Whitley [11] compared Lamarckian, Baldwinian and pure genetic algorithms in evolving the architecture and the weights of neural networks that learn Boolean functions. They conclude that using either form of leaning is better than using a pure genetic algorithm. Orvosh and Davis [55] found that 5% partial Lamarckian is the optimal learning strategy to solve the survival network design problem and the graph coloring problem. Michalewicz and Nazhiyath [109] replaced 20% of the repaired solutions in their hybrid algorithm to solve numerical optimization problems with nonlinear constraints. Bala et al. [110] showed how the Baldwin effect can improve the performance of a genetic algorithm when integrated with a decision tree in order to evolve useful subsets of discriminatory features for recognizing complex visual concepts. However, Ku and Mak [111] found that only using Lamarckian evolution improved the performance of genetic algorithm in evolving recurrent neural networks. They also concluded that effective hybridization depends on the local search method used and the learning frequency. Houck et al. [99] used seven problems to compare the performance of different learning strategies. Their investigation concluded that neither the pure Lamarckian nor pure Baldwinian strategy was found to be consistently effective. It was discovered that the 20% and 40% partial Lamarckian search strategies yielded the best mixture of solution quality and computational efficiency based on a minmax criterion (i.e. minimizing the worst case performance across all test problems instance). Sasaki and Tokoro [112] found that adaptation by Lamarckian evolution was much faster for neural networks than Darwinian evolution in a static environment. However, when the environment changed from generation to generation, the Darwinian evolution was superior. Julstrom [24] reported that Baldwinian strategies perform poorly in solving the 4-cycle problem compared to a pure genetic algorithm and their effectiveness deteriorates with an increasing use of learning in contrast to Lamarckian strategies. He also found that applying Lamarckian leaning to all the individuals produced the most effective results. Joines et al. [100] found that using the pure Lamarckian approach (100% Lamarckian) produced the best convergence speed to the best known solution when solving the cell formation problem. Espinoza et al. [112] used 75% partial Lamarckian as the optimal leaning strategy in their hybrid to optimize two continuous functions. El-Mihoub et al. [40] investigated the combined effect of probability of local search and leaning strategy on the hybrid performance and found that combing a low probability of local search with the pure Lamarckian learning strategy can improve the convergence speed without disrupting the schema processing. Ishibushi et al. [114] found that the 5% partial Lamarckian worked well on the multi-objective 0/1 knapsack problem using a single population model, however, the 50% partial Lamarckian was the optimal choice using the island model.

The effectiveness of adopting the pure Lamarckian approach, the pure Baldwinian approach, or any mixture of them in a hybrid is affected by the fitness landscape, the representations, the percentage of population performs local search and local search method used [40], [99], [100], [103], [109], [114].

B. Balance between Global and Local Search

The hybrid algorithm should strike a balance between exploration and exploitation, in order to be able to solve global optimization problems. According to the hybrid theory [115], solving an optimization problem and reaching a solution of desired quality can be attained in one of two ways. Either the global search method alone reaches the solution or the global search method guides the search to the basin of attraction from where the local search method can continue to lead to the desired solution. In the genetic-local hybrid, the main role of the genetic algorithm is to explore the search space in order to either isolate the most promising regions of the search space, or, to hit the global optimum. However, the main role of the local search method is to exploit the information gathered by the global genetic algorithm. The division of the hybrid's time between the two methods influences the efficiency and the effectiveness of the search process. The optimal division of

the algorithm's time is an important issue that is faced the designers of hybrid genetic algorithms.

Although the aim of combining a global genetic algorithm and a local search method is to get the best out of the exploring ability of the former, and the efficiency of the latter in reaching local optima, the two methods can interact in a more complicated way than the one described above. Rosin et al. [116] argued that the mutation operator in a hybrid plays a different role than it does in a pure genetic algorithm. The local refinement requirement of the mutation operator becomes unnecessary in the existence of an explicit local operator allowing the mutation operator to take a more exploratory role. Land [117] suggested using larger mutations, at least large enough to move from one basin to another, in cases where each individual of the population is completely locally optimized. He went further, when he argued that local search obviates the need for crossover in solving the graph bisection problem, because local search is able to build the very same building blocks that the crossover would otherwise combine.

The exploring ability of the genetic algorithm can be further improved by utilizing local search to ensure fair representation of different regions of a search. This can improve the ability of the genetic algorithm to direct the search to the most promising regions of the search space. Once the algorithm has guided the search to the basin of attraction of the global optimum, utilizing local search can further improve the search to produce an effective optimization algorithm. The first goal of the hybridization, which is the effectiveness of search, can be satisfied if a genetic algorithm and a local search method cooperate in the manner mentioned above. However, there are other more destructive forms of interaction. For example, the mutation and crossover operators can disrupt good and complete local solutions which may waste algorithm resources and produce an inefficient search. The Lamarckian local search can disrupt the schema processing of the genetic algorithm which may lead to premature convergence and produce an ineffective search.

In addition to the role of genetic operators in systemically exploring the search space, they perform some form of local search with relative low cost compared to the more accurate local search methods. The improper use of the expensive local search in a hybrid can waste algorithm resources. The algorithm should be able to decide wisely on both methods, especially when both can achieve the desired task, taking into account the benefits and costs of their utilization. The condition of an appropriate use of both methods in addition to the condition of interacting in a cooperative way should be satisfied in order to produce an effective and efficient search algorithm.

Researchers have proposed different techniques to enable the hybrid to mix both methods wisely or at least to reduce the consequences of the improper use of the expensive local search. These techniques are based on modifying the different parameters of a local search method within a hybrid. Modifying the parameters of the local search, such as the frequency of local search, the duration of local search, and the probability of local search can help the hybrid to strike the balance between the two search methods.

1) Frequency of Local Search

The number of continuous uninterrupted generations that a genetic algorithm performs before applying local search is usually referred to as the frequency of local search. In the traditional hybrid genetic algorithm, the frequency of local search is 1, for example. The staged hybrid genetic algorithm [118], [119] was designed to separate the two search methods into two distinct stages by increasing the frequency of the local search in order to minimize the interference between the two search methods. Mathias and Whitely [118] used a local search frequency of 2 to solve the TSP. However, in a hybrid algorithm to solve the static correction problem [119], the genetic search algorithm was allowed to continue uninterrupted for ten generations before applying a single iteration of waveform steepest ascent iteration to each individual in the population. This hybrid algorithm produced solutions with improved quality of 5% and additional savings in time compared with the traditional hybrid genetic algorithm. Espinoza et al. [113] conducted a set of experiments to find the optimal local search frequency for two two-dimensional continuous test functions and they found that the optimal frequency of local search for these test functions was 3.

The optimal frequency of local search is function dependent and varies with time because the optimal time that should be spent on local and global search algorithms depends on the distribution of individuals in the population. Syrjakow and Szczerbicka [120] studied the optimal switch point between the genetic algorithm and local search to fine-tune the solution found by the pre-optimizer genetic algorithm. They studied three criteria: the number of function evaluations, the convergence speed of the genetic algorithm, and the regional accumulation of search points indicating the convergence toward a specific region in the search space so as to determine the optimal switch point. The convergence speed criterion produced the highest efficiency in their experiment. Lobo and Goldberg [13] addressed the problem of deciding between global search and local search in order to make the most out of either technique. They tried to answer the question, "when should the local search be used and when should the global genetic algorithm be used to achieve the maximum possible efficiency?" They viewed the problem as a two armed bandit problem where the payoff of each bandit is unknown and changes with time. They presented a model for efficient hybridizing based on the concept of probability matching. This model can be viewed as an adaptive technique that adjusts the frequency of local search depending on the efficiency of both genetic and local techniques as the search progresses. Tuson and Ross [121] used a similar model to adapt the operator probability in their cost based operator rate adaptation. They used their model to select the use of a mutation or crossover operation in a pure genetic algorithm. The same technique has been used to solve the three-matching problem [52], where an adaptive hybrid algorithm selects one operator from eight recombination and local search operators based on their

current and past benefit-cost ratio.

Espinoza et al. [113] used the change in coefficient of variation of the fitness function to determine whether the genetic algorithm is exploring new regions of the search space or exploiting the already visited regions. Based on that, the algorithm selects to perform either a genetic or a local iteration. The algorithm relies on the local search role to improve the sampling of the new regions that are being explored in the case of any increase in that coefficient. Once the search has branched to a local search, the fitness improvement-cost ratio of both the last genetic and the local iterations and the maximum number of local iterations are used to decide on continuing the local search or going to the global search. Their experiments showed that the algorithm is more efficient than a pure genetic algorithm and is stable against a greater range of parameter settings than the standard staged hybrid genetic algorithm.

Hacker et al. [28] proposed an approach that switches between global genetic and local search, based on the local topology of the search space. The basic idea of this approach ignores the role of local search in improving the sampling ability of the genetic algorithm. It concentrates on the efficiency of local search, i.e. finding the optimum once the global genetic algorithm has defined its basin of attraction. The utilization of the relative homogeneity of the population and regression analysis to determine whether the search is exploring a single basin or multiple basins was investigated. The coefficient of variance of both the fitness and phenotype is used to quantify the relative homogeneity of the population. A decrease in the values of the coefficient of variance indicates that the genetic algorithm has converged to a small area of the search space and the search process can therefore be made more efficient by switching to a local search. In contrast, an increase in its value indicates that a new region of the search space is being explored and hence there is less need to use a local search. Regression analysis has also been used to determine when to switch between global and local techniques. The value of the error of fitting the population of solutions to a second-order surface can indicate whether the genetic algorithm is exploring multiple basins or a single basin in the search space. Depending on the value of that error, the algorithm decides to switch to a local search or continue the global search. They concluded that utilizing local search could be helpful for small search spaces in the early stages of search due to their role in helping the genetic algorithm to define the most promising regions of the search space. However, for large and complicated search spaces, their role is limited to accelerating finding of the global optimum once the genetic algorithm isolates the most promising region and can be helpful in later stages of the search.

2) Duration of Local Search

Local search duration influences the balance between the global exploration of genetic algorithms and local refinement of the neighborhood search method in hybrid genetic algorithms [122], [123]. A hybrid with long local search duration will execute fewer generations of the genetic algorithm than a hybrid with shorter local duration, if both terminate after the same number of function evaluations.

On combinatorial domains, a local search can be performed until a solution converges to a local optimum. However, on continuous domains, the local search is typically truncated before reaching a local optimum when its step length becomes too small. Performing local search until a solution converges to a local optimum, which is referred to as complete local search, may lead to the loss of population diversity [102] depending on the learning strategy used. Hybrid genetic algorithms that adopt the pure Lamarckian approach are more prone to loss of diversity than others which utilize other learning techniques.

Applying a complete local search on costly function evaluations can also be expensive. However, there is a certain class of problems, decomposable fitness problems [124], where calculating the fitness of a solution given the fitness of its neighbor, is significantly less computationally expensive than computing its fitness from scratch. TSP is an example of this group of problems where computing the length of a tour that shares most of its edges with another tour, whose length is already known, is much cheaper than computing the length of a complete tour. Radcliffe and Surry [124] argued that hybrids are more suitable for problems exhibiting this property.

A few studies have been conducted which investigate the optimal duration of local search. Hart [7] found that using a short duration of local search produced the best results for the Griewank functions [125], whereas a long duration produced better results for the Rastrigin functions [126]. Rosin et al. [116] experimented with very short and very long local search durations in a hybrid to optimize the drug-docking configuration. Both durations were found to yield similar performance. Hart et al. [122] concluded that duration of local search is an important factor and hybrid genetic algorithms with long local searches will be most effective for nontrivial problems.

The high cost of a complete local search on expensive function evaluations makes any improper use of the local search difficult to recover from. However, the recovering from any misuse of partial local search is still possible. Partial local search is more suitable for hybrids that decide on a global or a local approach depending on the current state of the search and the previous performance of both methods. In this case, where there is a possibility of misjudgment in some circumstances, the use of partial local search gives the hybrid a higher chance to recover from such errors than using a complete local search.

3) Probability and Selection of Local Search

In any hybrid algorithm, a local search can be applied to either every individual in the population or only few individuals. In traditional hybrid genetic algorithms, a local search is applied to every individual in the population. However, applying a local search to every individual in the population on costly function evaluations can waste resources without providing any more useful information. In this case, the local search can be applied to individuals that fall in the same basin of attraction of the search space, whereby producing the same local optimum. Applying a local search to

a large fraction of the population can limit exploration of the search space by allowing the genetic algorithm to evolve for a small number of generations. The possibility of applying local search on more than one individual from the same basin can be reduced by performing local search on only a small fraction of the population. This also lowers the chances of applying an unnecessary local search on individuals that fall in non-promising regions of the search space. Deciding upon the optimal fraction of the population which should perform local search, and the basis on which these individuals are chosen, has a great impact on the performance of a hybrid.

Hart [7] investigated the impact of the fraction of the population that undergo local search on the performance of real-coded genetic algorithms. He found that a relation exists between this fraction, the population size and the performance of the hybrid. He also found that performing local search on small fractions could be more efficient when using larger populations and those large fractions can help to reflect the search space characteristics when using small populations. He concluded that a more selective use of local search could improve the efficiency of hybrids. Hart and Belew [127] studied the impact of the local search probability on the efficiency of hybrids. Their studies indicate that the probability of local search should be kept low in the initial stages and incremented in later generations. The population diversity in the initial stages of genetic algorithm enables good sampling of the search space. However, as the diversity diminishes in the later stages, the sampling ability of the genetic algorithm requires additional help from the local search.

Different techniques, such as tuning, distribution-based [7], fitness-based [7] techniques, and local search potential [117], have been proposed to decide on the optimal fraction of the population that should perform a local search. These techniques aim to reduce unnecessary local searches. However, they differ in the way they select individuals that perform the local search.

a) Tuning Technique

In the tuning technique, a primary experiment is conducted in order to find the optimal fraction of the population that should perform local search. This fraction is usually referred to as the probability of local search. This value is then used to run the real experiment and remains fixed during the run. Typically, the individuals that undergo local search are chosen uniformly at random. Rosin et al. [116] applied local search to 7% of the population in each generation in their hybrid to solve the docking problem. In Land et al. [128], only 5% of randomly selected individuals of the population perform a Marquardt-Levenberg local search in their hybrid to determine the basic parameters that describe the structure of a semiconductor wafer. Hart et al. [122] and Morris et al. [17] applied local search to 6% of the population. Espinoza et al. [113] found applying local search on 10% of the population produces the best efficiency for both their adaptive hybrid algorithm and the standard staged hybrid algorithm. In their adaptive hybrid genetic algorithm, this value is used as an initial value for the probability of local search, which is reduced by a specific value after applying local search. In a hybrid to solve TSP, Krasnogor and Smith [32] applied their adaptive local search method with a probability of 1.0 to each individual in the population, except the one with the best fitness.

b) Distribution-based Technique

Distribution-based techniques modify the probability of local search based on the distribution of individuals in the population. The motivation for these techniques is to ensure that only one individual from each basin of attraction in the search space can undergo local search. These techniques can improve the sampling ability of the hybrid by preventing bad representatives of good regions from misguiding the global genetic algorithm.

Hart [7] used the F statistic as a measure of distance over the space of genotypes to adapt the probability of local search. Joines and Kay [107] combined evolutionary algorithms with random linkage and borrowed the concept of short memory from tabu search [129] to avoid performing unnecessary local search on non-promising regions of the search space. The authors defined tabu hyperspheres around the offspring of the genetic algorithm to reduce the number of wasted function evaluations owing to the rediscovery of the same local optimum. The probability of local search of each offspring depends on the distance to the nearest tabu region. By decreasing the size of these tabu hyperpheres as the search progress, the algorithm can intensively search the most promising regions of the search space. This in turn can help to find the exact local optimum of the region which also represents the global optimum of the search space. The authors compared their hybrid using the Lamarckian learning approach with a pure genetic algorithm, and the standard hybrid genetic algorithm where each offspring perform local search using two different learning strategies. They reported that their hybrid outperformed other algorithms in terms of both solution quality and computation effort. Martinez-Estudillo et al. [130] selected individuals for local search using clustering techniques to optimize the structure and the weights of product-unit based neural networks. The results showed that the clustering approach was able to perform better than similar algorithms that do not use clustering analysis.

c) Fitness-based Technique

A fitness-based technique adaptively calculates the probability with which local search is applied. This technique uses the fitness information in the population to bias the local search towards individuals that have a better fitness. The local search probability of each individual is modified based on the relationship of its fitness to the fitness of other individuals. These methods assume that individuals with better fitnesses are more likely to be in the basins of attraction of the most promising regions. This assumption ignores the dynamic of genetic algorithms and the cumulative effect of applying local search on successive generations which can aggravate the sampling ability of the global genetic algorithm and can

misguide the search. For example, if a promising region of the search space is represented poorly by an individual with under-average fitness and, in the same population, a non-promising region is represented by individuals with over-average fitness, the representative of the non-promising region will have more chance to perform local search and improve its chances of survive.

Hart [7] found no statistical differences between the results obtained by applying fitness-based selection and the results of fixed probability of local search. Espinoza et al. [131] used a clustering technique that is tailored to the three different stages the authors have defined for constrained problems to adapt the probability of local search. In the first stage, where all the solutions are infeasible, and the last stage, where all the solutions are feasible, the authors experimented with clustering the individuals depending on their fitness. The selection was performed by means of Latin-hypercube sampling from clusters which had formed. In the second stage, where a few individuals are feasible, the probability of local search is proportional to the number of feasible solutions in the population. The results showed that the algorithm, which is based on a fitness clustering technique, is more reliably faster than the adaptive hybrid genetic algorithm with fixed starting local search probability. Lozano et al. [132] proposed a simple adaptive scheme which sets the probability of local search of each individual to either 1.0 or 0.0625 depending on the individuals fitness compared to the fitness of the current worst individual in the population. The authors concluded that this adaptation mechanism allows the balance between the global genetic search and the local search to be adjusted according to the particularities of the search space, thus allowing significant improvements in the performance for different classes of problems.

d) Local Search Potential Technique

The local search (LS potential) potential selection mechanism has been proposed by Land [117] to decide which individuals should perform the local search. Land suggested that biasing the local search towards individuals that can be most efficiently improved by local methods makes the most effective use of local search. The least easily improved solutions are likely to be those at or near to the local optimum and it is inappropriate to expend effort on fine refinement, as long as there are large differences in the population's fitness. In this way, the scheme biases the hybrid towards more exploration. As the population gets closer to the optima, this mechanism allows local search to progress to the next level of refinement. In his algorithm, he used the past local search effectiveness as a measure to estimate future effectiveness.

Different techniques have been used to control the different parameters of the local search in order to strike a balance with the global genetic methods. Most of the controlling techniques which are described by Eiben et al. [17] for controlling the parameters of evolutionary algorithms have been applied to the local search control parameters in a hybrid.

The self-adaptation techniques are reported to be successfully used to decide between different local search methods in solving the OneMax problem, NK-Landscapes, and TSP [134]. The self-adaptation technique has also been used to adapt the duration of local search in a hybrid through encoding the number of local iterations into chromosomes [106]. In this way, the global genetic algorithm decides on the individuals that should perform a local search and on its duration.

IV. Summary

In this paper, we have tried to shed some light on the effectiveness and efficiency of hybridizing genetic algorithms with various techniques through reviewing some of the wide variety of hybrid genetic approaches. These approaches show that hybridizing is one possible way to build a competent genetic algorithm [135] that solves hard problems quickly, reliably and accurately without the need for any forms of human intervention. Hybridization has been utilized to construct competent genetic algorithms that belong to two of the three main approaches for building competent genetic algorithms, i.e., perturbation, linkage adaptation, and probabilistic model-building [136]. The collective learning genetic algorithm is an example of a competent genetic algorithm that employs specifically designed representation and operators for adapting genetic linkage along with the evolutionary process. Other search and optimization methods can also be used to adapt genetic linkage. Probabilistic Model-Building Genetic Algorithms (PMBGA) are examples of probabilistic model builders which learn genetic linkage via building models based on the current population.

Hybridization is also one of the four main techniques for efficiency enhancement of genetic algorithms. Hybridization can also be used as a tool to achieve evaluation relaxation, which in turn is another main technique for efficiency enhancement.

The ability of a genetic-local hybrid to solve hard problems quickly depends on the way of utilizing local search information and the mechanism of balancing genetic and local search. By reviewing the different hybrid approaches, some of the important factors that affect the hybrid performance have been presented. This review shows that there is a trend towards adapting some of the hybrid design choices through adapting the control parameters associated with these choices while the search is progressing. Different adaptation techniques have been used to adapt the selection of a local search method, the selection of individuals for a local search, the duration of local search, the learning strategy, and other design aspects.

References

[1] J. Holland, *Adaptation in Natural and Artificial Systems*: The University of Michigan, 1975.

[2] K. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Doctoral Dissertation. Ann Arbor: The University of Michigan, 1975.

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

[4] D. Beasley, D. R. Bull, R, and R. Martin, "An overview of genetic algorithms: part 1, fundamentals," University Computing, vol. 15, pp. 58-69, 1993.

[5] H. Asoh and H. Mühlenbein, "On the mean convergence time of evolutionary algorithms without selection and mutation," in Parallel Problem Solving from Nature, PPSN III, Y. Davidor, H.-P. Schwefel, and R. Manner, Eds. Berlin, Germany: Springer-Verlag, 1994, pp. 88–97.
[6] D. Thierens, D. Goldberg, and P. Guimaraes, "Domino convergence, drift, and the temporal-salience structure of problems," in 1998 IEEE International Conference on Evolutionary Computation Anchorage, USA: IEEE, 1998, pp. 535-540.
[7] W. E. Hart, "Adaptive global optimization with local search," Doctoral Dissertation. San Diego: University of California 1994.
[8] S. Mahfoud and D. Goldberg, "Parallel recombinative simulated annealing: a genetic algorithm," Parallel Computing, vol. 21, pp. 11-28, 1995.
[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671-680, 1983.
[10] S. W. Mahfoud, "Boltzmann selection," in Handbook of Evolutionary Computation, T. Back, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publising Ltd and Oxford University Press, 1997, pp. C2.5:1-4.
[11] F. Gruau and D. Whitley, "Adding learning to the cellular development of neural network: evolution and Baldwin effect," Evolutionary Computation, vol. 1, pp. 213-233, 1993.
[12] C. Reeves, "Genetic algorithms and neighbourhood search," in Evolutionary Computing, AISB Workshop, vol. 865 Lecture Notes in Computer Science, T. C. Fogarty, Ed. Leeds, UK: Springer-Verlag, 1994, pp. 115-130.
[13] F. G. Lobo and D. E. Goldberg, "Decision making in a hybrid genetic algorithm," in IEEE International Conference on evolutionary Computation. Piscataway, USA: IEEE Press, 1997, pp. 122-125.
[14] K. De Jong, "Genetic algorithms: a 30 year perspective," in Perspectives on Adaptation in Natural and Artificial Systems, L. Booker, S. Forrest, M. Mitchell, and R. Riolo, Eds.: Oxford University Press, 2005.
[15] P. Preux and E.-G. Talbi, "Towards hybrid evolutionary algorithms," International Transactions in Operational Research, vol. 6, pp. 557-570, 1999.
[16] K. Deb, "Limitations of evolutionary computation methods," in Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publishing and Oxford University Press, 1997, pp. B2.9.
[17] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," IEEE Transactions on Evolutionary Computation, vol. 3, pp. 124-141, 1999.
[18] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy and design issues," IEEE Transactions on Evolutionary Computation, vol. 9, pp. 474-488, 2005.
[19] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," California Institute of Technology 1989.
[20] E. Talbi, "A Taxonomy of hybrid metaheuristics," Journal of Heuristics, vol. 8, pp. 541–564, 2002.
[21] A. A. Hopgood, Intelligent Systems for Engineers and Scientists, 2nd ed: CRC Press, 2001.
[22] T. Yamada and C. Reeves, "Solving the C_{sum} permutation flowshop scheduling problem by genetic local search," in International Confrenence on Evolutionary Computation. Anchorage, USA, 1998, pp. 230-234.
[23] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function," Journal of Computational Chemistry, vol. 19, pp. 1639-1662, 1998.
[24] B. Julstrom, "Comparing Darwinian, Baldwinian, and Lamarckian search in a genetic algorithm for the 4-cycle problem," in the 1999 Genetic and Evolutionary Computation Conference, Late Breaking Papers, S. Brave and A. S. Wu, Eds. Orlando, USA, 1999, pp. 134-138.
[25] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs third ed: Springer-Verlag, 1996.
[26] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," in AISB Workshop on Evolutionary Computing, vol. 993, Lecture Notes In Computer Science, T. C. Fogarty, Ed. Sheffield, UK: Springer Verlag, 1995, pp. 25-39.
[27] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Politecnico di Milano, Milan 1991.
[28] K. A. Hacker, J. Eddy, and K. E. Lewis, "Efficient global optimization using hybrid genetic algorithms," presented at 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, USA, 2002.
[29] E. Cantú-Paz, "A survey of parallel genetic algorithms," Calculateurs Parallele, Reseaux et Systems Repartis, vol. 10, pp. 141-171, 1998.
[30] D. E. Goldberg, "Using time efficiently: genetic-evolutionary algorithms and the continuation problem," in the Genetic and Evolutionary Computation Conference. Orlando, USA, 1999, pp. 212-219.
[31] D. E. Goldberg, "Foreward," EURASIP Journal on Applied Signal Processing, vol. 8, pp. 731-732, 2003.
[32] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in the Genetic and Evolutionary Computation Conference. Las Vegas, USA Morgan Kaufmann, 2000, pp. 987–994.
[33] S. Areibi and A. Vannelli, "Advanced search techniques for circuit partitioning," in Quadratic Assignment and Related Problems, vol. 16, DIMACS series in Discrtete Mathematics and Theoretical Computer Science, P. Pardalos and H. Wolkowicz, Eds., 1994, pp. 77-98.
[34] E. Besnard, N. Cordier-Lallouet, A. Schmitz, O. Kural, and H. P. Chen, "Design/optimization with advanced simulated annealing," American Insitute of Aeronautic and Astronautics 1999.
[35] K. Liang, X. Yao, and C. Newton, "Combining landscape approximation and local search in global optimization," in the Congress on Evolutionary Computation, vol. 2. Washington DC, USA: IEEE Press, 1999, pp. 1514-1520.
[36] J. Yen, J. C. Liao, B. Lee, and D. Randolph, "A Hybrid approach to modeling metabolic systems using genetic algorithms and simplex method," IEEE Transactions on Systems, Man, and Cybernetics, vol. 28, pp. 173-191, 1998.
[37] M. Chen and Q. Lu, "A hybrid model based on genetic algorithm and ant colony algorithm," Journal of Information & Computational Science, vol. 2, pp. 647-653, 2005.
[38] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. l. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," Evolutionary Computation, vol. 7, pp. 231 - 253, 1999.
[39] F. B. Espinoza, B. Minsker, and D. Goldberg, "Performance evaluation and population size reduction for self adaptive hybrid genetic algorithm (SAHGA)," in the Genetic and Evolutionary Computation Conference, vol. 2723, Lecture Notes in Computer Science San Francisco, USA: Springer, 2003, pp. 922-933.
[40] T. El-Mihoub, A. Hopgood, L. Nolle, and A. Battersby, "Performance of hybrid genetic algorithms incorporating local search," in 18th European Simulation Multiconference (ESM2004), G. Horton, Ed. Magdeburg, Germany, 2004, pp. 154-160.
[41] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in the International Conference on Genetic Algorithms and their Applications. Hillsdale, USA: Lawrence Erlbaum, 1985, pp. 154-159.
[42] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. van Gucht, "Genetic algorithms for the traveling salesman problem," in the First International Conference on Genetic Algorithms and Their Applications, J. J. Grefenstette, Ed. Pittsburgh, USA: Lawrence Erlbaum, 1985, pp. 160-165.
[43] Y. Davidor, Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization: World Scientific Publishing, 1991.
[44] E. K. Burke, D. G. Elliman, and R. F. Weare, "A hybrid genetic algorithm for highly constrained timetabling problems," in the sixth International Conference on Genetic Algorithms, L. J. Eshelman, Ed. Pittsburgh, USA Morgan Kaufmann 1995, pp. 605-610.
[45] B. Freisleben and P. Merz, "New genetic local search operators for the travaling salesman problem," in the Fourth Conference on Parallel Problem Solving from Nature vol. 1141, Lectures Notes in Computer Science, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 890–899.
[46] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," Operations Research, vol. 21, pp. 498-516, 1973.
[47] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," in the Second International Conference on Genetic Algorithms on Genetic algorithms and their application. Hillsdale, USA, 1987, pp. 224 - 230.
[48] A. Homaifar, S. Guan, and G. E. Liepins, "Schema analysis of the traveling salesman problem using genetic algorithms," Complex Systems, vol. 6, pp. 533-552 1992.
[49] J. Wroblewski, "Theoretical foundations of order-based genetic algorithms," Fundamenta Informaticae, pp. 423–430, 1996.

[50] D. Whitley, T. Starkweather, and D. A. Fuquay, "Scheduling problems and traveling salesman: the genetic edge recombination operator," in the Third International Conference on Genetic Algorithms. Fairfax, USA, 1989, pp. 133 - 140.
[51] P. Jog, J. Y. Suh, and D. Van Gucht, "Parallel genetic algorithms applied to the traveling salesman problem," SIAM Journal of Optimization, vol. 1, pp. 515-529, 1991.
[52] G. Magyar, M. Johnsson, and O. Nevalainen, "An adaptive hybrid genetic algorithm for the three-matching problem," IEEE Transaction on Evolutionary Computation, vol. 4, pp. 135-146, 2000.
[53] B. Burdsall and C. Giraud-Carrier, "Evolving fuzzy prototypes for efficient data clustering," in Second International ICSC Symposium on Fuzzy Logic and Applications. Zurich, Switzerland, 1997, pp. 217-223.
[54] T. Ibaraki, "Combinations with other optimization methods," in Handbook of Evolutionary Computation, T. Back, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publishing and Oxford University Press, 1997, pp. D3:1.
[55] D. Orvosh and L. Davis, "Shall we repair? genetic algorithms, combinatorial optimization, and feasibility constraints," in the Fifth International Conference on Genetic Algorithms. Urbana-Champaign, USA: Morgan Kaufmann, 1993, pp. 650.
[56] J. Abela, D. Abramson, M. Krishnamoorthy, A. D. Selva, and G. Mills, "Computing optimal schedules for landing aircraft," in the 12th Conference of the Australian Society for Operations Research. Adelaide, 1993, pp. 71-90.
[57] A. Konak and A. E. Smith, "A hybrid genetic algorithm approach for backbone design of communication networks," in the 1999 Congress on Evolutionary Computation. Washington D.C, USA: IEEE, 1999, pp. 1817-1823.
[58] S. Areibi and Z. Yang, "Effective memetic algorithms for VLSI design = genetic algorithms + local search + multi-level clustering," Evolutionary Computation, vol. 12, pp. 327 -353 2004.
[59] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," Soft Computing, vol. 9, pp. 3-12, 2005.
[60] S. Lawrence, A. C. Tsoi, and A. D. Bäck, "Function approximation with neural networks and local methods: bias, variance and smoothness," in Australian Conference on Neural Networks. Canberra, 1996, pp. 16–21.
[61] L. Willmes, T. Bäck, Y. Jin, and B. Sendhoff., "Comparing neural networks and kriging for fitness approximation in evolutionary optimization," in IEEE Congress on Evolutionary Computation. Canberra, Australia, 2003, pp. 663-670.
[62] Y. Jin and B. Sendhoff, "Reducing fitness evaluations using clustering techniques and neural network ensembles," in Genetic and Evolutionary Computation Conference (GECCO 2004), vol. 3102 Lecture Notes in Computer Science. Seattle, USA: Springer, 2004, pp. 688-699.
[63] C. A. Ankenbrandt, B. Buckles, F. E. Petry, and M. Lybanon, "Ocean feature recognition using genetic algorithms with fuzzy fitness functions," in the Third Annual Workshop on Space Operations, Automation and Robotics. Houston, USA, 1989, pp. 679-685.
[64] R. Pearce and P. H. Cowley, "Use of fuzzy logic to describe constraints derived from engineering judgment in genetic algorithms," IEEE Transactions on Industrial Electronics, vol. 43, pp. 535-540, 1996.
[65] H.-P. Schwefel, "Advantages (and disadvantages) of evolutionary computation over other approaches," in Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publishing and Oxford University Press, 1997, pp. A1.3.
[66] M. Pelikan, D. E. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic models," IlliGAL 1999.
[67] S. Baluja, "Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon University 1994.
[68] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "BOA: the Bayesian optimization algorithm," in the Genetic and Evolutionary Computation Conference. Orlando, USA: Morgan Kaufmann, 1999, pp. 525-532.
[69] S. Tsutsui, M. Pelikan, and A. Ghosh, "Performance of aggregation phermone system on unimodal and multimodal problems," in The 2005 IEEE Congress on Evolutionary Computation, vol. 1. Edinburgh, UK: IEEE, 2005, pp. 880-887.
[70] L. T. Leng, "Guided genetic algorithm," Doctoral Dissertation. University of Essex, 1999.
[71] E. P. Tsang and C. Voudouris, "Fast local search and guided local search and their application to British telecom's workforce scheduling problem," In Operations Research Letters, vol. 20, pp. 119-127, 1997.
[72] A. Hedar and M. Fukushima, "Simplex coding genetic algorithm for the global optimization of nonlinear functions," in Multi-Objective Programming and Goal Programming, Advances in Soft Computing, T. Tanino, T. Tanaka, and M. Inuiguchi, Eds.: Springer-Verlag, 2003, pp. 135-140.
[73] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," IEEE Transactions On Evolutionary Computation, vol. 6, pp. 580- 593, 2002.
[74] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm with a new termination criterion, H_ϵ gate, and two-phase scheme," IEEE Transactions on Evolutionary Computation, vol. 8, pp. 156-169, 2004.
[75] H. Talbi, A. Draa, and M. Batouche, "A new quantum-inspired genetic algorithm for solving the travelling salesman problem," in 14th International Conference on Computer Theory and Applications. Alexandria, Egypt 2004.
[76] K. C. Tan, Y. Li, D. J. Murray-Smith, and K. C. Sharman, "System identification and linearisation using genetic algorithms with simulated annealing," in First IEE/IEEE Int. Conf. on GA in Eng. Syst.: Innovations and Appl. Sheffield, UK, 1995, pp. 164-69.
[77] T. Yamada and R. Nakano, "A genetic algorithm with multi-step crossover for job-shop scheduling problems," in First IEE/IEEE International Conference on Genetic ALgorithms in Engineering Systems Innovations and Applications (GALESIA '95): Sheffield, UK, 1995, pp. 146–151.
[78] T. P. Riopka and P. Bock, "Intelligent recombination using individual learning in a collective learning genetic algorithm," in the Genetic and Evolutionary Computation Conference (GECCO-2000), D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Las Vegas, USA: Morgan Kaufmann, 2000, pp. 104-111.
[79] F. Herrera and M. Lozano, "Heuristic crossovers for real-coded genetic algorithms based on fuzzy connectives," in the 4th International Conference on Parallel Problem Solving from Nature, vol. 1141, Lecture Notes In Computer Science. Berlin, Germany: Springer-Verlag 1996, pp. 336 - 345.
[80] J. N. Richter and D. Peak, "Fuzzy evolutionary cellular automata," in International Conference on Artificial Neural Networks in Engineering, vol. 12. Saint Louis, USA, 2002, pp. 185-191.
[81] F. Herrera and M. Lozano, "Adaptive genetic operators based on co-evolution with fuzzy behaviors," IEEE Transactions on Evolutionary Computation, vol. 5, pp. 149-165, 2001.
[82] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: using the genetic algorithm with connectionist learning," in Artificial Life II. New York, USA: Addison-Wesley, 1991, pp. 511-547.
[83] H. Liang, Z. Lin, and R. W. McCallum, "Application of combined genetic algorithms with cascade correlation to diagnosis of delayed gastric emptying from electrogastrograms," Medical Engineering & Physics, vol. 22, pp. 229–234, 2000.
[84] D. J. Montana, "Neural network weight selection using genetic algorithms," in Intelligent Hybrid Systems: John Wiley & Sons, 1995, pp. 85-104.
[85] P. Arena, R. Caponetto, I. Fortuna, and M. G. Xibilia, "MLP optimal topology via genetic algorithms," in the International Conference on Artificial Neural Nets and Genetic Algorithms, A. Dobnikar, N. Steele, D. Pearson, and R. F. Albrecht, Eds. Portoroz, Slovenia: Springer-Verlag, 1993, pp. 670-674.
[86] N. Chaiyaratana and A. M. Zalzala, "Hybridisation of neural networks and a genetic algorithm for friction compensation," in The 2000 Congress on Evolutionary Computation, vol. 1. San Diego, USA, 2000, pp. 22-29.
[87] J. R. Koza and J. P. Rice, "Genetic generation of both the weights and architecture for a neural network," in Joint Conference on Neural Networks, vol. 2. Seattle, USA, 1991, pp. 397-404.
[88] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in the Third International Conference on Genetic Algorithms, J. D. Schaffer, Ed. San Mateo, USA: Morgan Kaufmann, 1989, pp. 379-384.
[89] D. Chalmers, "The evolution of learning: an experiment in genetic connectionism," in Connectionist Models,1990 Summer School, D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, Eds. San Diego, USA: Morgan Kaufmann, 1990, pp. 81–90.
[90] J. Fontanari and R. Meir, "Evolving a learning algorithm for the binary perceptron," Network, vol. 2, pp. 353–359, 1991.
[91] X. Yao, "Evolving artificial neural networks," Proceedings of the IEEE, vol. 87, pp. 1423-1447, 1999.
[92] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in the Fourth International Conference on Genetic Algorithms. San Diego, USA: Morgan Kaufmann, 1991, pp. 450-457.

[93] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," IEEE Transaction on Fuzzy Systems, vol. 1, pp. 46-53, 1993.
[94] M. Valenzuela-Rendon, "The fuzzy classifier system: motivations and first results," in the International Workshop Parallel Problem, vol. 496, Lecture Notes in Computer Science. Dortmund, Germany: Springer, 1991, pp. 338-342.
[95] T. White, B. Pagurek, and F. Oppacher, " ASGA: improving the ant system by integration with genetic algorithms," in the third Conference on Genetic Programming (GP/SGA'98). Madison, USA, 1998, pp. 610-617.
[96] H. M. Botee and E. Bonabeau, "Evolving ant colony optimization," Advanced Complex Systems, vol. 1, pp. 149-159, 1998.
[97] M. L. Pilat and T. White, "Using genetic algorithms to optimize ACS-TSP," in the Third International Workshop on Ant Algorithms, vol. Lecture Notes In Computer Science 2463. Berlin, Germany: Springer-Verlag, 2002, pp. 282 - 287.
[98] G. Hinton and S. J. Nowlan, "How learning can guide evolution," Complex Systems, vol. 1, pp. 495-502., 1987.
[99] C. Houck, J. Joines, M. Kay, and J. Wilson, "Empirical investigation of the benefits of partial Lamarckianism," Evolutionary Computation, vol. 5, pp. 31- 60, 1997.
[100] J. A. Joines, M. G. Kay, R. King, and C. Culbreth, "A hybrid genetic algorithm for manufacturing cell design," Journal of the Chinese Institute of Industrial Engineers, vol. 17, pp. 549-564, 2000.
[101] C. Sung-Soon and M. Byung-Ro, "A graph-based Lamarckian-Baldwinian hybrid for the sorting network problem" IEEE Transactions on Evolutionary Computation, vol. 9, pp. 105- 114, 2005.
[102] D. Whitley, S. Gordon, and K. Mathias, "Lamarckian Evolution, the Baldwin effect and function optimization," in Parallel Problem Solving from Nature - PPSN III vol. 866, Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Manner, Eds. Jerusalem: Springer-Verlag, 1994, pp. 6-15.
[103] P. Turney, "Myths and legends of the Baldwin effect," in Workshop on Evolutionary Computation and Machine Learning at the 13th International Conference on Machine Learning. Bari, Italy, 1996, pp. 135-142.
[104] P. Turney, D. Whitley, and R. Anderson, "Evolution, learning, and instinct: 100 years of the Baldwin effect," Evolutionary Computation, vol. 4, pp. iv-viii, 1996.
[105] G. Mayley, "Landscapes, learning costs and genetic assimilation," Evolutionary Computation, vol. 4, pp. 213 - 234, 1996.
[106] T. El-Mihoub, A. Hopgood, L. Nolle, and A. Battersby, "A self-adaptive Baldwinian search in hybrid genetic algorithms," in the 6th Fuzzy Days International Conference on Computational Intelligence. Dortmund, Germany: Springer, 2006, to be published.
[107] J. A. Joines and M. G. Kay, "Hybrid genetic algorithms and random linkage," in the 2002 Congress on Evolutionary Computation. Honolulu, USA: IEEE, 2002, pp. 1733-1738.
[108] W. E. Hart, T. E. Kammeyer, and R. K. Belew, "The role of development in genetic algorithms," in the Third Workshop on Foundations of Genetic Algorithms. San Fransico, USA, 1995, pp. 315-332.
[109] Z. Michalewicz and G. Nazhiyath, "Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," in 2nd IEEE International Conference on Evolutionary Computation, vol. 2. Perth, Australia IEEE, 1995, pp. 647-651.
[110] J. Bala, K. A. D. Jong, J. Huang, H. Vafaie, and H. Wechsler, "Using learning to facilitate the evolution of features for recognizing visual concepts," Evolutionary Computation, vol. 4, pp. 297–311, 1996.
[111] K. W. Ku and M. W. Mak, "Exploring the effects of Lamarckian and Baldwinian learning in evolving neural networks," in International Conference on Evolutionary Computation. Indianapolis, USA, 1997, pp. 617-622.
[112] T. Sasaki and M. Tokoro, "Adaptation toward changing environments: why Darwinian in nature?," in Fourth European Conference on Artificial Life, , Complex Adaptive Systems Series P. Husbands and I. Harvey, Eds. Brighton, UK: MIT press, 1997, pp. 145-153.
[113] F. B. Espinoza, B. Minsker, and D. Goldberg, "A self adaptive hybrid genetic algorithm," in the Genetic and Evolutionary Computation Conference (GECCO 2001). San Francisco, USA: Morgan Kaufmann Publishers, 2001, pp. 759.
[114] H. Ishibuchi, S. Kaige, and K. Narukawa, "Comparison between Lamarckian and Baldwinian repair on multiobjective 0/1 knapsack problems," in Evolutionary Multi-Criterion Optimization, Carlos A. Coello Coello, A. H. Aguirre, and E. Zitzler, Eds. Guanajuato, Mexico, 2005, pp. 370-385.
[115] D. E. Goldberg and S. Voessner, "Optimizing global-local search hybrids," in the Genetic and Evolutionary Computation Conference (GECCO 1999). Orlando, USA: Morgan Kaufmann, 1999, pp. 222-228.
[116] C. D. Rosin, R. S. Halliday, W. E. Hart, and R. K. Belew, "A comparison of global and local search methods in drug docking," in the Seventh International Conference on Genetic Algorithms, T. Bäck, Ed. Michigan, USA: Morgan Kaufmann, 1997, pp. 221-228.
[117] M. Land, "Evolutionary algorithms with local search for combinatorial optimization," Doctoral Dissertation. San Diego: University of California 1998.
[118] K. Mathias and D. Whitley, "Genetic operators, the fitness landscape and the traveling salesman problem," in Parallel Problem Solving from Nature-PPSN 2. Brussels, Belguim: North Holland-Elsevier, 1992, pp. 219-228.
[119] K. Mathias, L. Whitley, C. Stock, and T. Kusuma, "Staged hybrid genetic search for seismic data imaging," in International Conference on Evolutionary Computation. Orlando, USA, 1994, pp. 356-361.
[120] M. Syrjakow and H. Szczerbicka, "Combination of direct global and local optimization methods," in IEEE Conference on Evolutionary Computation. Perth, Western Australia: IEEE, 1995, pp. 326-333.
[121] A. L. Tuson and P. Ross, "Cost based operator rate adaptation: an investigation," in the Fourth International Conference on Parallel Problem Solving From Nature (PPSN IV), Lecture Notes in Computer Science. Berlin, Germany: Springer Verlag, 1996, pp. 461-469.
[122] W. E. Hart, C. R. Rosin, R. K. Belew, and G. M. Morris, "Improved evolutionary hybrids for flexible ligand docking in AutoDock," in Optimization in Computational Chemistry and Molecular Biology, C. A. Floudas and P. M. Pardalos, Eds.: Springer 2000, pp. 209-230.
[123] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," IEEE Transactions on Evolutionary Computation, , vol. 7, pp. 204- 223, 2003.
[124] N. J. Radcliffe and P. D. Surry, "Formal memetic algorithms," in Evolutionary Computing: AISB Workshop. Brighton, UK: Springer-Verlag, 1994, pp. 1-16.
[125] A. O. Griewank, "Generalized descent for global optimization," Journal of Optimization Theory and Applications, vol. 34, pp. 11-39, 1981.
[126] A. Törn and A. Zilinskas, "Global optimization," in Lecture Notes in Computer Science, vol. 350: Springer-Verlag, 1989.
[127] W. E. Hart and R. K. Belew, "Optimization with genetic algorithm hybrids that use local search," in Adaptive individuals in evolving populations: Models and algorithms, vol. 26, R. Belew and M. Mitchell, Eds.: Addison-Wesley, 1996, pp. 483-496.
[128] M. Land, J. J. SIDorowich, and R. K. Belew, "Using genetic algorithms with local search for thin film metrology," in the Seventh International Conference on Genetic Algorithms. East Lansing, USA: Morgan Kaufmann, 1997, pp. 537-544.
[129] F. Glover, "Tabu search- part I," ORSA Journal on Computing, vol. 1, pp. 190-260, 1989.
[130] A. Martinez-Estudillo, C. Hervas-Martnez, F. Martnez-Estudillo, and N. Garca-Pedrajas, "Hybrid method based on clustering for evolutionary algorithms with local search," IEEE Transactions on Systems, Man and Cybernetics, 2004.
[131] F. Espinoza, B. S. Minsker, and D. Goldberg, "Local search issues for the appliction of a self-adaptive hybrid genetic algorithm in groundwater remediation design," in American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) World Water & Environmental Resources Congress 2003 & Related Symposia. Philadelphia, USA, 2003.
[132] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," Evolutionary computation, vol. 12, pp. 273 - 302 2004.
[133] F. T. Lin, C. Y. Kao, and C. C. Hsu., " Incorporating genetic algorithms into simulated annealing," in the Fourth International Symposium on Artificial Intelligence. Cancun, Mexico, 1991, pp. 290-297.
[134] N. Krasnogor and J. Simth, "Emergence of profitable search strategies based on a simple inheritance mechanism," in the Genetic and Evolutionary Computation Conference. San Francisco, USA: Morgan Kaufmann, 2001, pp. 432-439.
[135] D. Goldberg, "The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity," in Evolutionary Design by Computers: Morgan Kaufmann, 1999, pp. 105–118.

[136] Y. Chen and D. Goldberg, "Convergence time for the linkage learning genetic algorithms," Evolutionary computation, vol. 13, pp. 279-302, 2005.

Tarek A. El-Mihoub graduated with a BSc in computer engineering from Al-Fateh Uinversity, Tripoli, Libya in 1993 and obtained his MSc in engineering multimedia from Nottingham Trent University in UK by the end of 2002.

He is currently a PhD Student at Nottingham Trent University. He worked as a Teaching Assistant at Al-Fatah University in Libya and as a Manager of computer department of the Libyan environment general authority. His current research is in the field of optimization, genetic algorithms, and artificial intelligence.

Adrian A. Hopgood graduated with a BSc (Hons) in physics from the University of Bristol in 1981 and obtained a PhD from the University of Oxford in 1984.

He is professor of Computing and Dean of the School of Computing & Informatics at Nottingham Trent University, UK. He is also a visiting professor at the Open University. His main research interests are in intelligent systems and their practical applications.

Prof. Hopgood is a fellow of the British Computer Society and a committee member for its specialist group on artificial intelligence.

Lars Nolle graduated from the University of Applied Science and Arts in Hanover in 1995 with a degree in Computer Science and Electronics. After receiving his PhD in Applied Computational Intelligence from The Open University, he worked as a System Engineer for EDS.

He returned to The Open University as a Research Fellow in 2000. He joined The Nottingham Trent University as a Senior Lecturer in Computing in February 2002. His research interests include: applied computational intelligence, distributed systems, expert systems, optimization and control of technical processes.

Alan Battersby obtained an MSc in Computer Science from Hatfield Polytechnic, UK in 1977.

Prior to joining the School of Computing and Informatics at Nottingham Trent University, UK, he was a Computing Development Officer for Bedfordshire Education Authority. His research interests include: Fuzzy Logic applied to Robotics, wavelets, compression and the Internet.