Dynamic Problems and Nature Inspired Meta-heuristics

Tim Hendtlass[1], Irene Moser[1], and Marcus Randall[2]

Abstract Biological systems have often been used as the inspiration for search techniques to solve continuous and discrete combinatorial optimisation problems. One of the key aspects of biological systems is their ability to adapt to changing environmental conditions. Yet, biologically inspired optimisation techniques are mostly used to solve static problems (problems that do not change while they are being solved) rather than their dynamic counterparts. This is mainly due to the fact that the incorporation of temporal search control is a challenging task. Recently, however, a greater body of work has been completed on enhanced versions of these biologically inspired meta-heuristics, particularly genetic algorithms, ant colony optimisation, particle swarm optimisation and extremal optimisation, so as to allow them to solve dynamic optimisation problems. This survey chapter examines representative works and methodologies of these techniques on this important class of problems.

1 Introduction

Many industrial optimisation problems are solved in environments that undergo continual change. These problems are referred to as *dynamic optimisation problems* and are characterised by an initial problem definition and a series of 'events' that change it over time. An event defines some change

Centre for Information Technology Research
School of Information and Communication Technologies
Swinburne University, VIC 3001
{thendtlass, imoser}@ict.swin.edu.au
School of Information Technology
Bond University, QLD 4229
Australia
mrandall@bond.edu.au

A. Lewis et al. (Eds.): Biologically-Inspired Optimisation Methods, SCI 210, pp. 79–109.
springerlink.com

either to the data of the problem or its structural definition *while the problem is being solved*. In comparison to static optimisation problems, dynamic optimisation problems often lack well defined objective functions, test data sets, criteria for comparing solutions and standard formulations [7, 37].

Evolutionary algorithms are those based on natural and biological systems. A very common example of these are genetic algorithms (GAs). For this class of algorithms, extensive modifications to accommodate dynamic optimisation problems have been made. A survey of these approaches up to the year 2000 is given by Branke [18]. However, for another group of evolutionary algorithms, *Ant Colony Optimisation* (ACO) [35], *Particle Swarm Optimisation* (PSO) [39] and *Extremal Optimisation* (EO) [13], suitable modifications and applications to these difficult problems are only starting to appear.

This chapter presents a survey of representative GA, ACO, PSO and EO works and methodologies for dynamic problems. Sections 2 (post 2000), 3, 4 and 5 describe how each of the identified meta-heuristics has been used to solve dynamic optimisation problems. Section 6 comments on the limitations of dynamic techniques.

2 Genetic Algorithms

Genetic Algorithms are generally regarded as the classical biologically inspired optimisation algorithms. Hence they are the predominant representatives of stochastic approaches in the work of researchers over the years, and they also dominate the field of approaches to solving dynamic problems. The concept of GAs was devised by Holland [53]. In essence, each solution to a problem is treated as a chromosome, a population of which are used in the system. Each solution component of the problem under consideration is referred to as a gene. In early GAs, these were nearly exclusively composed of sequences of bits called alleles, though real numbers and integers are now commonly used instead. At each iteration of the algorithm, a subset of the fittest chromosomes (i.e., those having the best objective function values) are permitted to form the basis of the next generation. This is often done by use of the crossover operator, in which gene segments of one individual are replaced by segments of another thus forming a child chromosome. This can be repeated until a new population is formed. Mutation of some gene values also occurs to increase the genetic diversity of the population. Over time, increasingly fitter individuals are produced, though premature convergence may limit the extent of the optimality of the chromosome solutions.

There are numerous adaptations to dynamic optimisation problems and the applications designed until the year 2000 have been reviewed in-depth by Branke [18]. The GA adaptations discussed are mainly concerned with postponing convergence to retain the ability to search the problem space sufficiently widely after a change. The initial approaches up to around the

year 2000 include memorising individuals [61, 74], adjusting the mutation rate [1, 24, 25], redundant representation [43, 50, 69], randomisation [44], fitness sharing [60] and multiple populations [18, 88].

2.1 Memory

Explicit memory as a GA variation was first introduced by Ramsey and Grefenstette [74] as a knowledge base of memorised individuals, devised as an approach to detecting and exploiting similarities between problem states. When a problem enters a new state, the similarities with previous states are determined and individuals which are well adapted to these states are reintroduced into the population.

A substantial contribution on the usage of memory in connection with GAs solving dynamic problems has been made by Branke [16]. The initial paper explores a memory-based approach which divides the real-value encoded population into subpopulations. Initially, the population is subdivided into two parts, one of which is randomised after a change. This part of the population contributes individuals to the memory while the other part is responsible for the search, for which it retrieves individuals from the repository. In a second alternative, the population is split into three islands with all populations contributing to the memory while only one of the populations retrieves individuals from this memory. The individuals in the memory are counted as part of the population, which has a constant size. Individuals are retrieved from memory after each detected change. Only the best individual is added to memory from time to time (in this case, every ten generations), and several replacement strategies are explored. The two best-performing strategies propose to replace the most similar individual, alternatively to replace the most similar individual only if it is of worse fitness. The experiments on the Moving Peaks problem, using five peaks, compare the performance of the implementations with one, two and three populations combined with memory as well as a single-population GA without memory. The results show a superior performance of the implementation with two populations when the peaks change their heights but do not move, whereas the three-population variation performs best among the candidates when the peaks change their locations as well as their heights. Branke [16] concedes that memory-based approaches have limited application since they are only truly useful in the case of an exact recurrence of a previous situation but also concludes that increased diversity contributes more to the performance of a memory-based GA than to a standard implementation.

Bendtsen and Krink [8] argue that repetitive patterns are typical for real problems in everyday life, as various scheduling problems typically adhere to daily or weekly patterns. The dynamic memory they employ does not store the generation-best individual directly, but approaches it in real-valued

space. The memorised individual closest to the best in the main population is moved towards this best individual by a Gaussian random number with a mean of 0.5. At every generation, the best individual from memory replaces the worst individual in the main population. Individuals in memory that do not have the best individual as a closest neighbour are moved the same distance in a random direction. The authors report that their memory-enhanced GA outperforms the implementation by Branke [16] as well as the standard GA used as a reference. As test cases, they used the moving peaks function developed by Morrison and De Jong [65] and a greenhouse problem.

Karaman, Uyar and Eryiğit [57] employ a memory which characterises the environment as it presents itself just before a change. A key figure defined as the ratio of feasible individuals over all possible individuals identifies the environment, which is then reproduced probabilistically using a frequency array of alleles (essentially a vector of the distributions of ones on the allele locations of all individuals). When the algorithm is notified of a change, it records the current individuals in the distribution array before the individuals are replaced by the closest matching environment. To establish the closest recorded environment, 100 random locations are specified at the beginning of the trial and evaluated after each change according to the new objective function. If no sufficiently close environment is available, the algorithm uses hypermutation to facilitate quick adaptation. The authors explore the algorithm ('MIA') by experimenting on a knapsack problem with 50 different environments (changes). Both a GA with hypermutation and a GA with memory are used as a benchmark. The authors conclude that the devised MIA algorithm performs better the longer the algorithms run, achieving better results after the initial 1000 generations.

Inspired by Ramsey and Grefenstette [74] as well as Karaman et al. [57] Yang [93] further explored the use of memorised individuals which provide information regarding the environment they were initially stored from. The Associative Memory GA (AMGA) algorithm updates its memory, whose size is one tenth of the main population, by storing the best individual every 5–10 generations along with a frequency array of alleles as in Karaman et al. [57]. The individual and its distribution vector replace the closest individual in memory. After a change, the fittest individual in memory is chosen and a definable number of individuals are created from the distribution vector to represent a previous environment. All new individuals are merged with the population and the worst individuals are then discarded until the original population size is restored. The experiments reveal that this algorithm works best when the number of new individuals introduced equals approximately 50% of the size of the main population. Also, AMGA works best when solving cyclic problems in which previous environments recur. It solves problems with random changes to better quality than the standard GA and the memory-based GA. The latter which merges all of the memorised individuals into the main population and keeps the best.

2.2 Memory and Randomisation

Trojanowski and Michalewicz [87] are the first authors cited for combining the memory and randomisation techniques with GAs. Replacing large parts of the population with random individuals after a change and keeping an individual memory are methods of adaptation which are explored separately and in combination. The individual memory is implemented as a memory buffer each individual updates after a generation. In the case of mutation, the previous chromosome is added to the memory buffer; in the case of crossover, the better parent's chromosome history is recorded in the buffer. When the buffer is full, the oldest individual(s) are replaced. After a change, all individuals are re-evaluated, including the chromosomes in the memory buffers. If one of the memorised individuals has a better fitness, it replaces the currently active individual. When randomisation is used in combination with memory, the memory is applied first. After this, some of the active individuals are re-randomised whereas the chromosomes in the buffers remain unaffected.

This approach was tested on a multimodal function using binary-encoded individuals with some cyclic and some random aspects. The authors observed that randomisation works best on small changes or a relatively easy objective function. They conclude that the combination algorithm finds the most optima in the multimodal function, with both the randomised and the memory-based variation outperforming the benchmark GA with standard settings.

Yang [92] used a memory whose size is one tenth of the number of individuals in the main population. After the initial random members populating the memory have been replaced with the main population's best individual every 5–10 generations, the individual closest to the replacing is substituted. If a change has taken place, the best individuals in memory are reintroduced into the main population. Yang proposes to combine this type of memory with random immigrants to be added to the main population every generation. Inspired by Bendtsen and Krink [8], Yang proposes a third variation in which the randomisation and the memory are combined. This algorithm, MIGA, makes random mutations to the best individual in memory instead of introducing random individuals. It is tested on several functions and is found to perform better than the other two variations as well as the existing random immigrants GA introduced by Cobb and Grefenstette [25].

Simões and Costa [78] have explored the use of memory as well as replacement strategies extensively. Initially, they hybridised a GA with two aspects from Immune Systems, clonal selection and hypermutation. Clonal selection is applied to the main population at every generation. The cloned individuals have a preset probability of being subjected to hypermutation, for which gene libraries, initialised randomly and kept constant throughout the generations, are used. A memory population is kept, in which the best individuals from different phases are recorded in combination with the average fitness of the main population at the time they were stored. When changes are

detected, the individuals with the closest average fitness number are cloned and reintroduced into the main population to replace the worst individuals.

This algorithm, named Immune System based GA (ISGA), was developed further by Yang [94], who introduced an alignment variable for the gene segments in the library. The goal is to enforce the use of the gene segments in a predefined location of the individual. The library has two kinds of gene segments: A random group to be re-randomised at every generation and a non-random one which contains copies of gene segments and their locations from members of the main population selected by a tournament of two. For the experiments, Yang's [94] dynamic test functions are used. The performance of the devised algorithm is superseded only by a version which uses non-aligned transformation, as in the algorithm of Simões and Costa [78]. Unfortunately, the algorithm is not compared to the original ISGA [78], but to another variation devised by the same authors.

In further work [79, 80, 81, 82], Simões and Costa devised and explored the Variable-size Memory Evolutionary Algorithm (VMEA), which is based on adapting the size of the main and memory populations. However, the memory and main populations only shrink when duplicates are removed. As the populations are often at a maximum, the replacement schemes used are of considerable importance. Two aging schemes and one generational method are proposed. The first aging scheme ages the individuals in memory linearly at every generation. The age is increased more considerably when the individual is reintroduced into the main population and reset to zero when a certain age limit is reached. When a replacement takes place, the youngest individual is removed. In the second aging scheme, devised originally by Branke [16], the individual's fitness contributes to its age as well as the generations. Although the age is never reset in this model, the youngest individual is replaced. The generational scheme prescribes that the worst individual added since the last change is replaced. If no individual has been added, the closest individual by Hamming distance is replaced. The authors found that the generational scheme outperforms the age-based schemes as well as the similarity-based replacement scheme found to be superior in Branke's studies [16].

The second age-based scheme was observed to perform well on problems with frequent changes. Apart from the memory updating schemes, the VMEA algorithm also explores a genetic operator which draws its inspiration from bacterial conjugation. It resembles two-point crossover, except that the gene sequence is copied from the better individual (the donor) to the individual with worse fitness (the recipient) so that the donor remains unchanged. In one variation of the VMEA algorithm, it is used in the place of the crossover operator. In most experiments, conjugation performed better than the crossover operator.

2.3 Randomisation

Motivated by the observation that completely random 'random immigrants' may interfere with the solution process when changes are not very severe, Yang [95] designed a GA in which the randomisation of individuals is based on mutations of the current elite. As expected, this algorithm outperforms a GA with 'random immigrants' on a dynamic environment with minor (20% and less) changes. Yang [95] goes on to show that if the goal is to cater for both severe and slight dynamics, a combination of elite-based and completely random individuals is a useful choice.

To further explore the potential of randomised elite individuals, introduced by Yang [95], Yang and Tinós [96] devised an algorithm combining three degrees of randomisation. Based on the observation that elite-based individuals are most useful only in an environment undergoing only minor changes, they combine the elite-based individuals with completely random as well as dual individuals. The latter describe the mirror image of existing elite individuals and are considered the appropriate response to the most drastic changes, while completely random individuals are considered the ideal response to changes of intermediate severity. All three groups of immigrant individuals are used in combination and replace the worst individuals after a change. The less successful of the three groups have their numbers reduced until a given limit is reached. Success is measured as the performance of the best individual of each group. The tests [96] demonstrate that a traditional GA with random immigrants is indeed less successful when the changes have a low severity of less than 20% of the environment. The hybrid algorithm performs best when the severity of changes fluctuates, as it uses different types of randomness adaptively.

2.4 Self-Organised Criticality (SOC)

Tinós and Yang [85] have devised a hybrid approach which combines both the randomisation of chosen individuals and the maintenance of a subpopulation. To introduce self-organised criticality [4], their method replaces both neighbours of the worst individual as well as the worst individual itself with random individuals. The neighbours are defined as adjacent entries in the array of individuals. If the new worst individual is not within index range of the previous subpopulation, the old subpopulation is replaced. If it is at the edge of the previous range, the subpopulation will grow, a phenomenon the authors call an extinction event. The new randomised individuals are kept as a separate subpopulation to ensure that they are not eliminated for poor performance before having been given a chance to evolve separately from the other individuals.

The subpopulation is more likely to grow when the general level of fitness in the population is high and the worst individual is likely to be a random individual introduced in the previous generation. If this worst individual has a member of the main population as a neighbour, the neighbour is also replaced, leading to randomisation of a larger neighbourhood. This emulates the SOC paradigm of inducing a critical state when the general level of adaptedness is high. The authors compare this approach to a standard GA and two implementations of a GA with random immigrants, the exact implementations of which are not discussed. The benchmark problems used are dynamic unitation functions, devised by the authors as 'deceptive' functions with local optima. The authors use the adaptability metric introduced by De Jong [26], and find that their hybrid approach outperforms the benchmarks. More extensive tests on the algorithm are presented in further work [86], where the algorithm is referred to as Self-Organising Random Immigrants GA (SORIGA).

2.5 Recent Applications

Chaudhry and Luo [22] provide a survey of GA applications to operational problems published in production and operations management journals in the years 1990–2001. They found that the number of publications is generally rising over the years, with 50% of the GA implementations applied to scheduling problems. The authors do not specify if any of these articles, 178 in total over the years, describe applications to dynamic problems.

Freight train scheduling is the optimisation problem for a two-phase GA application devised by by Godwin, Gopalan and Narendran [42]. In most countries, the freight train system uses the passenger rail network with passenger trains dictating the framework for the possible schedules. The rail network is subdivided into 'blocks', which are railway sections between possible sidings, where trains can pass each other. Locomotives and freight rakes – a set of connected freight carriages without a locomotive – are located within divisions or subsections of the railway network. Relocating a locomotive between divisions means an 'empty' trip and is known as 'deadheading'. The problem has two conflicting objectives, the minimisation of deadheading and the minimisation of coupling delay, effectively the time freight rakes wait for a locomotive.

The authors propose to address the problem in two phases; the first assigns the locomotive and does a preliminary scheduling which places the freight train between the passenger trains; the second phase minimises the arrival time at the destination and deals with conflicting freight train and deadheading schedulings. Both phases use a GA with single-point and two-point crossover, mutation and elitism as well as a local search procedure which explores the immediate neighbourhood of a solution by swapping genes. The

performance of the algorithm is not compared with other approaches but it is stated that, on the three hypothetical rail networks used for the experiment, the algorithm produced acceptable schedules.

The problem of inventory management is solved as a strategy problem with sliding time windows by Bosman and La Poutré [15]. The optimisation focuses on strategies to adopt when reordering and determining the size of the orders to place. These are referred to as the reorder point and order-up-to size respectively and are applicable to each of the available suppliers. Variations in lead and delivery times render the problem more difficult. As the decisions prescribed by the strategies influence the succeeding time windows, the authors chose to solve this problem on line with sliding time windows to continuously evolve suitable strategies for different situations. The dynamics are created by the fluctuating lead and delivery times.

The solver optimises the profit, which is a function of the sales revenue decreased by the holding cost and the ordering cost. The more reliable suppliers are assumed to be more expensive. If the stocks fall below a limit, emergency orders have to be placed with more expensive suppliers. The experiments use four scenarios in which the algorithm learns the optimal ordering strategy and clearly improves at doing this over time. In the fourth scenario, the authors introduce the anticipation of 'customer satisfaction' as an increase in the number of sales which depends on the inventory levels. Although an interesting consideration, it may not be a very realistic assumption.

3 Ant Colony Optimisation

ACO is an optimisation paradigm encompassing a range of meta-heuristics based on the evolutionary mechanics of natural ant colonies. Unlike many other biologically inspired optimisation algorithms, ACO, on the whole, represents a set of constructive techniques. This means that each ant at each step of the generalised algorithm adds a component value (such as the next city for the travelling salesman problem (TSP)) to its solution. This decision is based on a combination of the worth/goodness of component values and simulated chemical markers called pheromone. The use of the latter is a collective form of self adaptation. As a result, colonies of ants produce increasingly better solutions over time. Generally, pheromone is used to reinforce good solutions and their component values. However, there is also the capacity to decay pheromone strength, so that the effect of premature convergence is lessened. A comprehensive overview of ACO can be found in Dorigo and Di Caro [35].

As a maturing meta-heuristic, applications to dynamic problems have started to emerge. Apart from the benchmark problem set (including the TSP, quadratic assignment problem (QAP) and knapsack problem), industrial oriented research has been mainly in networks and telecommunications [35] and manufacturing. Scheduling, autonomous robots, continuous optimisation and

more generic frameworks for ACO in dynamic environments are also discussed.

3.1 Benchmark Problems

Benchmark combinatorial optimisation problems, such as TSP, QAP and the knapsack problem, are usually presented as static problems. However, as there has been recent interest in adapting ACO to solve dynamic problems, some common problems have been changed so they have a dynamic (temporal) component. In particular, this has been done for the TSP. There are two broad ways in which a standard TSP can be transformed into a dynamic problem. These are: dynamically varying the distances between cities; and adding/dropping cities from the problem while it is being solved between separate runs of the ant colony solver.

Eyckelhof and Snoek [40] solve problems in which the distances between cities vary dynamically. They adapt the basic Ant System algorithm to avoid the twin problems of a) certain edges becoming relatively unused because of low pheromone values and b) some edges having too much pheromone and hence dominating the decision process. This is achieved by effectively limiting the lower and upper bounds on pheromone values, implicitly implementing a form of $\mathcal{MAX} - \mathcal{MIN}$ Ant System. Using two small test data sets of 25 and 100 cities, they found that the algorithm is able to quickly adapt to the changing environment and is an effective solution method for these problems.

In contrast, Angus and Hendtlass [2] solve a problem in which cities are added or dropped rather than dynamically changing distances. The addition and removal of cities occurs in separate runs of the solver. This work showed that the ant algorithm could adapt quickly to the new problem scenario. Upon the removal or addition of a city, the pheromone levels on the problem edges were normalised.

A more intricate approach is that by Guntsch, Middendorf and Schmeck [48]. Groups of cities are replaced by new cities at different frequencies. This differs from two of the authors' previous work [45] which only considered a single change to a problem instance. The approach they use for this more complex version of the problem is twofold – using pheromone modification mechanisms and a heuristic called *KeepElite* for modifying solutions to be suitable for new problem definitions. In terms of the former, three strategies were trialled which variously modified pheromone values by resetting them to the initial value or adjusting them in proportion to either the distance or existing pheromone level from the newly inserted/deleted city. It was found that a combination of resetting the pheromone and proportional adjustment worked best. This combined with the *KeepElite* heuristic had better performance than using the pheromone adjustment strategies in their pure form.

3.2 Network and Telecommunications Problems

Solving dynamic network and telecommunication problems, which include such diverse areas as routing telephone calls and the regulation of network traffic flow, have been the subject of most research.

Schoonderwoerd, Holland, Bruten and Rothcrantz [77] use a highly abstracted ant based system for solving problems involving the symmetric forward routing of telephone calls between origin and destination nodes using a switched network. The networks they study are not capable of connecting all calls at a given time so the objective is to minimise the number of lost calls. In their ant based solution, ants operate independently of the calls. Each ant travels between an origin and a random destination node based on a function of pheromone, distance and node congestion. Calls are routed according to the pheromone level on each neighbouring node. The link with the maximum pheromone is always chosen. This approach compares favourably with existing mobile agent methods of British Telecom [77]. Heusse, Snyers, Guérin and Knutz [52] extended Schoonderwoerd et al.'s [77] work to asymmetric networks that are capable of dealing with changing bandwidth and topology requirements.

Saleh and Ghani [76] create a system called the For/Backward approach which combines epochal updates (i.e., updates of routing tables only after an ant has moved forward and backward through the network) and incremental updating (in addition, ants also update the tables as they traverse the network). Applying this to a simulation of the Malaysian backbone network showed that a small packet delay and a greater bandwidth throughput could be achieved.

The problem solved in Xia, Chen and Meng [91] is one in which a composition of web services, that themselves are composed of component services, is required. Each service has a quality of service (QoS) component and different users have different preference weightings on each QoS. The objective is to maximise users' satisfaction with the supplied services. The graph that the ants walk is formed from the services and changes as new services are added, old ones removed and QoS values change. A multiple pheromone strategy embedded in a dynamic ACO, where different pheromones tracked different QoS attributes, was used. In comparison to a GA and a standard single pheromone ACO, the authors' modified algorithm provided better performance on a single simulation problem.

AntHocNet [34] is designed specifically to route packets in mobile ad-hoc networks and is an updated version of the work in Di Caro, Ducatelle and Gambardella [33] and Ducatelle, Di Caro and Gambardella [38]. Two kinds of ants known as forward and backward ants are used to construct the initial routing tables. The former are used to locate all the practicable paths within the network while the latter move backwards from the destination and establish the paths. Operationally, data packets are routed stochastically and link failures may occur. In order to manage the dynamic nature of the network,

another set of ants, called proactive forward ants are used to maintain and improve the proactivity of paths. The authors note that while AntHocNet has very good performance in terms of its ability to successfully route packets, its computational overheads are more than other algorithms (such as the simpler relative routing algorithm of Perkins and Royer [73]). This is mainly due to the number of different ants.

Submarinian, Druschel and Chen [83] present two ant colony algorithms to solve a network routing problem in which network topology and link cost changes occur dynamically and unpredictably. Their first algorithm uses a simple shortest path approach and assumes symmetric path costs. Their second algorithm is a more robust version which is capable of multi-path planning and does not require symmetric path costs. Each algorithm uses slightly different ants. The first is considered standard as the ants will likely follow good non-congested routes as given by the forwarding tables. The ants of the second algorithm effectively ignore the forwarding tables and all paths have equal probability. In addition to having ants that move forward from source to destination, another set of ants are sent backward from the destination node to the source node with the aim of exploring the network as well as the routing tables of the routers that they encounter. Both algorithms perform better than standard approaches when there are higher network failure rates.

Di Caro and Dorigo [27, 28, 30, 29, 31] have designed an ant colony system (ACS) to build forwarding tables for packet routing problems (such as those that occur on the Internet). This system is known as AntNet. Like other approaches, the algorithm consists of two types of ants, the forward ants and the backward ants. Each forward ant constructs a path from a source node to its destination nodes and collects information about about the time it takes to move from node to node. The corresponding backward ant then moves from the destination to the source node, updating the routing tables of each of the nodes it visits. The authors have shown that AntNet compares favourably with existing routing algorithms. Zhang, Khun and Fromherz [98] have made substantial adaptations to the system to accommodate ad-hoc wireless sensor networks.

Di Caro and Dorigo [32] apply two versions of AntNet to the problem of adaptive learning in dynamic datagram networks. A datagram network is considered as an unreliable network as the user is not notified if the delivery of a packet fails. The first version of AntNet is denoted AntNet-CL and is the canonical form of the algorithm while AntNet-CO is specifically adapted to best-effort routing in datagram networks. Using simulations of heavy traffic networks, the results showed that AntNet-CO lead AntNet-CL in terms of performance (i.e., throughput, packet delay and resource utilisation). Both algorithms were better able to handle these conditions compared to five standard Internet routing algorithms.

Similar work to AntNet has also been undertaken by White, Pagurek and Oppacher [90] except that separate ant colonies are used to determine the

routes, allocate traffic to the routes and deallocate routes. Preliminary work has also been carried out on routing with fibre optic cables [89].

3.3 Industrial Manufacturing

Some of the most common problems in industrial manufacturing are job shop scheduling and machine sequencing.

Cicirello and Smith [23] solve a dynamic problem of routing jobs on a shop floor using an algorithm based on ACO principles. In their approach, each job is assigned to an ant that makes the routing decisions through the various machines on the shop floor. Pheromone is deposited on the route that each ant/job takes. This was shown to effectively balance the workload of the factory machines.

Aydin and Öztemel [3] describe a system for solving dynamic job shop sequencing problems using intelligent agents. In this paper, an agent does not solve a complete problem, but simply reacts to the requests from a simulated environment and develops an appropriate job priority list. There are two stages; a learning stage and a production stage. In the learning stage, the simulated environment gives the agent feedback about the performance which it then uses as a part of a reinforcement learning program. Arrival times and processing durations on particular machines for particular jobs are given by an exponential distribution. However, machine breakdowns and other events (such as rush orders) are not dealt with by this approach.

3.4 Scheduling

Gutjahr and Rauner [49] present an ACO algorithm to solve a regional nurse scheduling/rostering problem for the consolidated hospital system in Vienna, Austria. This is modelled as a constraint satisfaction problem that takes nurse preferred working days, working patterns, employee satisfaction and hospital costs into consideration. The hard constraints (such as the legal number of hours a nurse can work each week) must be satisfied within the system, whereas the soft constraints do not. The dynamic components of the problem come about as a result of different day-to-day requirements and the necessity of the system to discourage delaying resource-based decisions until the last possible day. The latter is achieved by using an urgency penalty in the problem model. Using a simulated environment, it was shown that ACO could deliver schedules that incurred significantly less operational costs and higher nurse satisfaction than those that could be produced by a greedy algorithm.

3.5 Autonomous Robots

In many ways, this category is an extension to scheduling and network problems. Detection of targets in a dynamic environment is a problem considered by Meng, Kazeem and Muller [64]. A hybrid ACO/PSO algorithm is developed for the control of many small scale robots (the number of which may itself vary dynamically). In essence, robots communicate with one another using virtual pheromone, but may only do so if they are in a permissible range. Each robot holds its own pheromone matrix that it modifies over time. The swarm intelligence component of the hybrid is concerned with optimally organising the behaviour of the robots at a global level. These concepts working in combination allow agents to make decisions about the direction in which they move. The factors taken into consideration include each target's utility value (e.g., attractiveness and distance separation), and the agent's inertia, experience, and interactions with other agents. Simulation results showed that the hybrid ACO/PSO outperformed, and was more consistent than, a standard ACO implementation.

The work of Pekala and Schuster [72] differs from the above as a heterogeneous swarm of robots is controlled by the ACO algorithm. The aim of the research is to 'develop a method or algorithm that can control a heterogeneous swarm in an optimal manner to accomplish any particular task without human intervention' (p. 354). Each agent/ant represents a robot and attempts to complete a set of jobs which then makes up a task. The different types of robots in the swarm can complete only certain job types. One of the distinguishing factors of the algorithm is its ability for robots to learn the order in which tasks should be completed using a quality matrix metric. The dynamic element of their Modified Dynamic ACO (MDA) allows the addition and deletion of jobs. The simulation environment used to test MDA models the scenario in which food must be brought to a kitchen and then stacked. In this application, robots may scout for food, transport it to the kitchen, or stack the food items. The simulation showed that the system's ability to learn appropriate job orders made it highly effective for this problem. In addition, partially disabled robots were still able to make an effective contribution to solving the problem.

3.6 Continuous Optimisation

Beyond discrete dynamic optimisation problems, ACO has also been applied to continuous problems. These have generally been solved by genetic algorithms and particle swarm optimisation. However, Dreo and Siarry [36] propose the hybrid algorithm *Dynamic Hybrid Continuous Interacting Ant Colony (DHCIAC)*. In addition to pheromone, ants also pass messages about search progress directly to any of the other ants in the colony. The authors

implemented two versions of DHCIAC that are differentiated by the local search algorithm. $\text{DHCIAC}_{\text{find}}$ uses the Nelder-Mead simplex algorithm while $\text{DHCIAC}_{\text{track}}$ uses a modified simplex algorithm known as dynamic simplex. These effectively implement diversification and intensification strategies respectively. It was found that $\text{DHCIAC}_{\text{track}}$ was better for problems that had a faster rate of change. Overall, DHCIAC generally appears to be useful for slow changing problems with many local optima – though comparative results with other algorithms were not provided.

3.7 General Approaches

There have only been limited attempts to modify standard ACO algorithms to process dynamic problems more seamlessly.

Population ACO (P-ACO) [46, 47] is an ACO strategy that is capable of processing dynamic optimisation problems. It achieves this by using a different pheromone updating strategy. Only a set of elite solutions are used as part of the pheromone updating rules. At each iteration, one solution leaves the population and a new one (from the current iteration) enters. The candidate solution to be removed can be selected on age, quality, a probability function or a combination of these factors. The authors argue that this arrangement lends itself to dynamic optimisation, as extensive adjustments, due to the problem change, need not be made to the pheromone matrix. Instead, a solution modified to suit the new problem is used to compute the new pheromone information. This modification process works for full solutions only and is tailored for particular problems. Experimental work on the dynamic TSP and QAP showed that P-ACO could adapt to small changes in the problem better than restarting the entire algorithm.

A set of generic modifications have been proposed to allow ACO to solve dynamic optimisation problems [75]. This framework defines categories of events that change the problem definition (i.e., data and/or structure) while ACO solves it. Rather than discarding solutions and restarting the construction process, if an event occurs, the process of deconstruction begins. Deconstruction removes the components from a solution that make it infeasible to the changed problem. Results for a dynamic version of the multidimensional knapsack problem showed that the modified ACO could quickly adapt to the problem changes. Further work on the dynamic aircraft landing problem [41] (using a real-time simulator) indicates that the approach is capable of producing good schedules in a timely fashion.

4 Particle Swarm Optimisation

The classical particle swarm optimisation algorithm [58] was inspired by the swarming behaviour of birds and fish. While the metaphor is a dynamic one, there have been few applications to dynamic problems. PSO tries to mimic the natural behaviour and applies it to a number of particles moving through problem space. This movement occurs under the influence of a number of factors, some of which can be considered personal in that no other particle is involved while others are social and involve more than one particle.

When the velocity of each particle is updated, the particle keeps a part of its velocity at the last iteration. The fraction that is kept, referred to as the momentum (or alternately as the inertial weight) of the particle, prevents any drastic velocity changes and may permit the particle to pass through a local optimum. This is clearly a personal factor. A further possible personal influence is a tendency to return to the best position yet found by this particle (*pbest*).

Apart from the personal there are also social influences. The particle is attracted towards the best position currently experienced by other particles that forms its local neighbourhood (*lbest*) and/or towards the best position found by any particle in the swarm so far (*gbest*). The neighbourhood of a particle can be defined by those other particles with some distance D of this particle. Alternatively, since all particle must have an identifying index, a neighbourhood can be defined as all particles whose indices are within a user specified number of this particles index. Since the particles in a neighbourhood move closer together these two approaches tend to merge after a period of time.

Each particle updates its velocity simultaneously, normally using some combination of personal and social influences. Momentum is almost always used and generally two others, at least one of which must be social. Using *pbest* and *lbest* (a less greedy form) encourages the parallel exploration of multiple local optima while using *gbest* and *lbest* (a more greedy form) encourages the whole swarm to converge on the best optimum encountered. Using *pbest* and *gbest* is also a viable option as experience shows that the number of particles that constitute a local neighbourhood is not critical and may be reduced to one. It is the interplay of the chosen influences which produces an efficient search mechanism.

4.1 Adapting PSO for Dynamic Problems

There are a number of non-biological adaptations that need to be made to the classical swarm algorithm so that it suits dynamic problems. These can be summarised as: preventing the complete convergence of the swarm, keeping personal and social reference points up to date and maintaining or generating

explorer particles far from any current point of convergence. Approaches that achieve at least one of these aims will be considered.

4.1.1 Preventing Total Convergence

Social influences between particles, attractions to *gbest* and *lbest*, will tend to result in total convergence. To change this it is necessary to introduce some counter influence. One method [10] is to give at least some particles a charge so that, by analogy with electrostatics, two particles experience a repulsive force as they approach and the swarm would then not be able to fully converge. The particles would in time reach some (possibly dynamic) equilibrium between the convergence and divergence effects, but this does not mean that they are actively exploring. A second method [19] is to divide the swarm into sub-swarms so that not all particles converge on the same point. A particle and its closest neighbours may form a sub-swarm if the variance in the fitness of the particles is less than some threshold. Any particle that is not a member of a sub-swarm belongs to the main swarm. These sub-swarms may merge, acquire extra particles from the main swarm or collapse back into the main swarm. Whilst developed for multi-modal functions this niching behaviour could also be used, in principle, to limit total swarm convergence. However the algorithm depends on a uniform distribution of particles in the search space, a condition that may be able to be met after initialisation but which is not met after convergence into the sub-swarms has taken place. An alternative approach to niching is described in Bird and Li [9].

4.1.2 Refreshing the Best Positions

If an attraction to *pbest* is being used these best positions may be updated by allowing particles to replace their previous best position with the current position periodically [20]. Choosing a suitable period without detailed knowledge of the problem being optimised can be problematic. If an attraction to *gbest* is being used, the fitness at this position may be periodically re-evaluated [54]. As the fitness at that point deteriorates, the probability that it will be replaced by another position as a result of the current fitness at that position increases. Again a suitable re-calculation frequency has to be chosen.

4.1.3 Forcing Explorer Particles

The simplest approach just requires that a number of particles be periodically moved to randomly chosen points and have their fitness re-evaluated [21]. Another approach organises particles in a tree with each particle being influenced

by the particle above it (social) and itself (best position and momentum). A particle swaps with the one above it if it out performs it. This gives a dynamic neighbourhood that does require extensive calculation. This has been adapted to dynamic problems by Janson and Middendorf [55, 56]. After the value of the best-known position (*gbest*) changes (it is re-evaluated every cycle) a few sub-swarms are reinitialised while the rest are reset (have their old personal best information erased and replaced with the current position). The sub-swarms then search for the new optimum. Blackwell and Branke [11] introduce a more elaborate approach using quantum particles. Using an analogy to quantum mechanics, a particle on measurement is placed randomly within a given radius of its net current point of attraction. A uniform distribution is used and a function chosen so that a finite probability exists of a movement to a distance far from the point of attraction.

4.1.4 Meeting All Three Requirements

The approaches described above could, if used in combination, be used to track dynamic problems. However, one further approach (WoSP) [51] has the ability to meet all three requirements simultaneously by altering the normal PSO requirement to inhibit total convergence to one that reinforces the tendency to totally converge.

The approach was originally developed to sequentially explore an arbitrarily large number of optima. An extra short-range force of attraction was added to the basic swarm equation. As a result of the discrete way in which fitness evaluations and updates to the velocity of the particles is done, an aliasing effect causes pairs of particles to approach, pass each other and then continue on at very high velocities. The probability that this will happen increases with decreasing distance between the particles. Particles are most likely to be at close range when the swarm converges. There is no longer a need to stop the particles fully converging. As the velocity with which the particles leave the swarm is variable, exploration can continue both locally and at a distance. The total swarm is automatically broken into a number of sub-swarms called waves, each with its own *gbest* value. Particles that leave a converging swarm as a result of this aliasing effect leave the wave they were in and join the highest numbered wave (creating a new one if no higher numbered wave exists). A form of evolution takes place with lower performing waves being compulsorily recruited into better performing higher numbered waves. Waves that run out of particles (owing to promotion or recruitment) die out. In this way there is a continual automatic updating of best position information available to the successive waves.

The main difference between the algorithm for static and dynamic problems is that in the former each particle keeps a tabu list of places that it has already explored and was repelled from any place on its tabu list. In this way re-exploration of any point in problem space is largely (but not totally)

eliminated. For dynamic problems this tabu list can be completely removed on the grounds that any particular point in problem space may be a good optimum at several disjointed times. Alternatively extending the idea from Janson and Middendorf [56], each of these previously explored optima could be periodically re-examined and only those points whose fitness had significantly changed are removed from the tabu lists. It is not clear at this stage how the evolutionary pressure that is an important part of WoSP would respond to dynamic problems. Clearly if a number of waves were generated and completed their existence (in the time it took for the dynamic problem to change significantly), evolution would have time to make a contribution. Should the problem change much faster than this, it is likely that the contribution evolution would be able to make would be, at best, limited.

Another integrated approach, the Unified Particle Swarm Optimising scheme (UPSO) was proposed for static environments by Parsopoulos and Vrahatis [70]. It is based on a PSO with constriction factor which uses both a global and a local best. The velocities for these two search directions are unified into a single direction balanced by a factor $0 \leq u \leq 1$. The combined velocity is formed using a weighting u for the global best and $(1 - u)$ for the local best. The scheme also proposes the contribution of a Gaussian random displacement to either the local or the global best particle as they form the combined velocity. The authors observe a natural balance between exploration and exploitation in this algorithm, suggesting it may resist premature convergence. Parsopoulos and Vrahatis [71] therefore conducted some experiments on UPSO in a dynamic environment. Five well-known function optimisation problems, the Sphere, Rosenbrock, Rastrigin, Griewank and Schaffers functions are adapted to exhibit a global optimum with Gaussian random fluctuations. The results were measured as the mean of the best positions over the iterations. The authors find that their approach, when used with $u = 0.2$ or $u = 0.5$ performs significantly better than algorithms which use the influence of either the global or the local best particles exclusively (i.e., when u is set to 1.0 or 0.0 respectively).

4.2 Some Industrial Applications

As is evident in the previous discussion, a relatively large amount has described how PSO may be adapted so that it is generally suited to dynamic optimisation problems. On the whole, far fewer works have yet been devoted to applying PSO beyond benchmark test functions. Some novel applications are described below.

Using PSO to optimise the performance characteristics of wireless local area networks has been undertaken by Kókai, Christ and Frühauf [59]. Their work showed that an adaptive PSO implementation, based on the framework of Hu and Eberhart [54], could outperform a genetic algorithm and

neighbourhood search in a simulation environment. The simulator modelled participants in the network shifting positions, failing transmitters and competition amongst transmitters. The system was able to adapt to these changes in the specified time interval of 1 ms so that communications would not break off. An additional feature of this work was that the PSO was directly implemented in Field Programmable Gate Array (FPGA) hardware.

In a different area of communications, Zhang, Li, Zhou, Shen, Zhang, Zhang, Wu, Yuan, Chen, Zhang, Yao and Yang [97] develop a two stage process for adaptive compensation of polarisation mode dispersion used in communication along fibre optic cables. The degree of polarisation in these links changes dynamically because of factors such as environmental temperature. This in turn affects the search space of the problem. Once a global optimisation phase of the problem is complete, the extent of the change is analysed and may invoke the tracking algorithm. This algorithm uses PSO to search around the previously found global best solution so as to track changing optima. From this the authors found that the response time for adapting to disturbances in the system was in the order of a few milliseconds – a good result for this application.

Muller, Monson and Seppi [68] turn their attention to an economic problem of pricing items in a dynamic and noisy environment, such as that which occurs in e-commerce. The aim is to maximise product prices in the midst of fluctuating demand. The authors propose a form of PSO, called P-Best Decay PSO (PBDPSO). Rather than resetting particles and losing valuable information about the search space, a decay function is used so that particles are attracted to new areas of space, despite the noise in the system. In a dynamic pricing simulation, this PSO compares very well to a the Kalman Filter, an approach traditionally used for this problem.

5 Extremal Optimisation

The paradigm of self-organised criticality [4] explains a wide range of natural dynamical systems and has been previously mentioned in relation to genetic algorithms. EO [12, 13, 14] uses elements of SOC [4] by replacing the worst individual (in this case a solution component) in a population by a random one. Over a number of iterations it is expected that the overall solution quality will increase. The original version mutated the worst component only. The absence of noise made the solver very deterministic and vulnerable to local minima. To counteract this, τ-EO was introduced. It assigns each solution component a rank k based on its current fitness within the solution and mutates it according to a probability distribution of $k^{-\tau}$.

Only a few attempts to apply EO to dynamic problems have been made. These use EO's flexible solving mechanism to adapt to underlying fluctuations automatically. EO is guided only by the component fitnesses, which

are associated with the objective function. Typically EO algorithms will incorporate subtle changes automatically, as long as they are reflected in the amended objective function.

5.1 The Satisfiability Problem

Menai [63] investigates the use of EO on Incremental Satisfiability (ISAT) problems. Two different types of dynamics are added to static satisfiability problems from the SATLIB library: ISAT1 is obtained by starting from one clause and adding the others one by one, ISAT2 divides the same standard problems into two equal parts and solves the first half before adding the second half of the clauses. Adding the dynamic aspect to the problem seems to increase the challenge for the EO solver. A comparison between the results of EO solving a problem with more dynamics (ISAT1) and its application to a problem with a single change (ISAT2) corroborates this assumption.

The performance of EO is compared with a variation of the WalkSAT algorithm described in detail in McAllister, Sellman and Krautz [62]. Unfortunately, the benchmark algorithm (called R-Novelty) is only applied to the static case of the 29 problems used. This is somewhat surprising as the algorithm is very similar to EO and can be adapted to dynamic problems as easily as EO. Compared to R-Novelty, EO has a better average success rate when solving ISAT2. Solving ISAT2, EO uses approximately as many iterations as R-Novelty takes to solve the static problem.

5.2 A Defense Application

The Swedish Defense Research Agency investigated target recognition in which increasing numbers of moving sensors produce reports that may or may not contain information on the targets. Fast optimisation techniques are necessary to cluster the incoming reports according to the objects, before the relevant information on target objects can be processed by a tracker module. Svenson [84] compares EO and τ-EO on this task, which requires the ability to include bursts of incoming reports whenever they arrive.

The experiment on clustering the incoming reports using the EO algorithm stops short of adding reports dynamically. It only observes that EO performs better with $\tau = 1.5$ than when setting $\tau = \infty$, and that the pairwise comparison of a smaller subset of records to establish the current fitness of the report within a cluster leads to a better result than the evaluation of a larger subset of pairs of reports.

5.3 Dynamic Composition Problem

In order to test the EO solver's capabilities at solving dynamic problems, Moser [66] especially designed the composition problem as a dynamic knapsack-like combinatorial problem, where items of different types have to be added to a solution until a predefined limit is reached. The item's cost attribute relates to the solution's capacity limit, whereas the value or benefit attribute contributes to the solution's quality which is maximised.

The EO solver optimises the quality by improving an initial random solution. A uniformly random component is picked for removal from the solution and replaced with a component not currently in the solution which is chosen according to the power-law distribution.

The problem is made dynamic by introducing various changes, such as changing the values of the cost and benefit attributes of some components or by removing some of the available items from the problem space. Other problem instances prescribe the change of the percentage of a type of item that has to be represented.

The experiments confirm that the iterative aspect of the EO solver has certain disadvantages – large jumps in the search space are virtually impossible as the number of possible deteriorating moves is limited. However, the EO solver is able to adapt to a change very quickly, producing good quality solutions within few iterations. Sometimes a change of the underlying problem helps the solver produce a better solution, as it enables a substantial change in the composition of the solution, which could not have been achieved by the iterative algorithm itself. Given the simplicity of the algorithm, it scales well with the number of components. It is able to handle solutions with over a hundred components having a thousand components available.

5.4 'Moving Peaks' Dynamic Function Optimisation

Dynamic function optimisation was explored by Moser [66] using the moving peaks benchmark function [17]. The benchmark consists of several scenarios in which randomly placed peaks move in a multi-dimensional landscape. The peaks change locations as well as height and width around a given average. For the scenarios, exact specifications exist and therefore, comparable results from different approaches are available.

Several solution representations were attempted, leading to different ways of traversing the search space. Due to the nature of the moving peaks landscape, more than one EO solution was employed to keep track of the peaks moving in the landscape. Nonetheless, the results received from the best-performing EO implementation provided no match for some of the PSO solvers. Further experiments showed that the canonical EO implementation is not well suited to the moving peaks landscape. It does not make use of the

exploitable features of the function. For example, it does not climb to the top of the hill it found and thus scores suboptimally on the standard performance measure, the offline error.

5.5 Aircraft Landing

The single-runway dynamic aircraft landing problem (ALP) consists of a sliding time window with aircraft reporting to the air traffic control for landing. According to the formulation by Beasley [6], the aircraft report target landing times. The problem quality is measured according to the penalties for aircraft landing before or after the target times. Optimising the schedule of a sequence of aircraft is straightforward once the order of aircraft is known. Realistically, there are not many aircraft in the active time window at any given time. The problem instances provided as benchmarks by Beasley [5] have window lengths between 2 and 18 aircraft.

The EO adaptation devised by Moser [67] treats the ALP as a permutation problem of the aircraft in the current time window. The EO solver produces candidates for a new solution using pairwise swaps of aircraft. All candidates have their schedules optimised deterministically before their costs are evaluated. The new solution is then chosen among the candidates. The new solution is compared to the current best-known solution. Aircraft are removed from the active time window when they are within a given distance from their scheduled landing times.

The EO solver produced ideal schedules for each individual time window in all cases in which the optimal schedule was known. Speeding up the arrival of aircraft to shorten the intervals to one fifth did not change the results.

6 Tracking Frequency Limitations

The previous sections have demonstrated that all four algorithms have the capacity to solve dynamic optimisation problems. This last part of this chapter considers what determines the maximum rate at which changes can occur before the algorithm performance essentially is reduced to that of random search.

For population-based algorithms that use a history of previous performance to guide future search, such as ACO, GAs and PSO, the obvious limitation comes from how fast they can learn to disregard the now irrelevant part of their history. For population-based ACO this either implies a short history list or a method of detecting and removing all historic paths that are no longer valid. For non-population based ACO this will require careful handling of the evaporation to deposition ratios in the time immediately

after a change has been detected. For PSO either the *gbest* value needs to be periodically re-evaluated and a reset of it and all *lbest* (or *pbest*) positions made as soon as *gbest* alters or, alternatively a sequence of sets of *gbest* and *lbest* (or *pbest*) values have to be used (as in WoSP [51]) so that changes can be responded to without having to be explicitly detected.

For the single individual algorithm EO there is no explicit historical information used to guide future exploration[1]. Instead the algorithm will detect a change when the current solution suddenly becomes invalid. The current stored best solution then has to be cancelled and a special repair procedure will then have to be performed on the current individual. This will take a number of changes and after this enough time must be allowed so that a good solution can be built up.

It is possible to describe the factors that will determine the time it will take any of these algorithms to respond to a change and again have a good valid solution. However, the stochastic nature of the algorithms (amongst other factors) makes it impossible to quantify what this delay will be, thus allowing the maximum rate of problem change to be specified. Therefore, some limiting rate must exist, albeit it problem and algorithm dependent. Should any of these algorithms be used on a problem changing faster that the relevant limiting rate, the effect will be, that the algorithm makes too infrequent a sampling of problem space. By analogy to the aliasing effect seen when a digital data stream is sampled below the Nyquist-Shannon frequency when high frequencies appear as lower frequencies, it can be predicted that the algorithms may well present solutions to problems that, in fact, have never been posed. Worse than getting wrong results, there may be no clear indication that this has most undesirable effect has occurred. This suggests that either practical experimentation involving varying the frequency of known problems, or more ideally an analytical analysis, should be undertaken so that an estimate of the limiting rates of change the algorithms can handle can be established.

Until this is done, since the limiting change rate for the four algorithms discussed in this chapter are almost certain not to be the same, one pragmatic solution may be to run two or more of the algorithms in parallel, only accepting the solutions when they are at least similar. However, in the long run this is hardly practicable and no substitute for a fuller understanding of how these algorithms work and thus of their limiting features. For all the progress made so far in applying these algorithms to dynamic problems much yet remains to be done.

[1] As a result of the fitness build up and collapse cycle inherent in the behaviour of EO, it may be that the currently best known solution was found some time ago and could thus be considered historic. However, since the development is unaware of this best value and always modifies the current individual there is no historic information used as there is with ACO, GAs and PSO.

References

[1] Angeline, P.: Tracking extrema in dynamic environments. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 335–345. Springer, Heidelberg (1997)

[2] Angus, D., Hendtlass, T.: Ant Colony Optimisation Applied to a Dynamically Changing Problem. In: Hendtlass, T., Ali, M. (eds.) IEA/AIE 2002. LNCS (LNAI), vol. 2358, pp. 618–627. Springer, Heidelberg (2002)

[3] Aydin, M., Öztemel, E.: Dynamic job shop scheduling using reinforcement learning agents. Robotics and Autonomous Systems 33, 169–178 (2000)

[4] Bak, P., Tang, C., Wiesenfeld, K.: Self-organized criticality: An explanation of 1/f noise. Physical Review Letters 59, 381–384 (1987)

[5] Beasley, J.: OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society 41(11), 1069–1072 (1990)

[6] Beasley, J., Krishnamoorthy, M., Sharaiha, Y., Abramson, D.: Scheduling aircraft landings – the static case. Transportation Science 34, 180–197 (2000)

[7] Beasley, J., Krishnamoorthy, M., Sharaiha, Y., Abramson, D.: The displacement problem and dynamically scheduling aircraft landings. Journal of the Operational Research Society 55, 54–64 (2004)

[8] Bendtsen, C., Krink, T.: Dynamic memory model for non-stationary optimisation. In: Proceedings of the Congress on Evolutionary Computation, pp. 992–997 (2002)

[9] Bird, S., Li, X.: Adaptively choosing niching parameters in a PSO. In: Proceedings of the 8th Conference on Genetic and Evolutionary Computation, pp. 3–10 (2006)

[10] Blackwell, T., Bentley, P.: Dynamic search with charged swarms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19–26 (2002)

[11] Blackwell, T., Branke, J.: Multi-swarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation 10, 459–472 (2006)

[12] Boettcher, S., Percus, A.: Extremal optimization: Methods derived from co-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 825–832. Morgan Kaufmann, San Francisco (1999)

[13] Boettcher, S., Percus, A.: Nature's way of optimizing. Artificial Intelligence 119, 275–286 (2000)

[14] Boettcher, S., Percus, A.: Optimization with extremal dynamics. Physical Review Letters 86, 5211–5214 (2001)

[15] Bosman, P., La Poutré, H.: Inventory management and the impact of anticipation in evolutionary stochastic online dynamic optimization. In: Proceedings of the Congress on Evolutionary Computation, pp. 268–275. IEEE Computer Society Press, Los Alamitos (2007)

[16] Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the Congress on Evolutionary Computation, pp. 6–9. IEEE Computer Society Press, Los Alamitos (1999)

[17] Branke, J.: The moving peaks benchmark (1999), `http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/`

[18] Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell (2001)

[19] Brits, R., Englebrecht, A., van der Bergh, F.: A niching particle swarm optimiser. In: Proceedings of the Asia Pacific Conference on Simulated Evolution and Learning, pp. 692–696 (2002)
[20] Carlisle, A., Dozier, G.: Adapting particle swarm optimization to dynamic environments. In: Proceedings of the International Conference on Artificial Intelligence, pp. 429–434 (2000)
[21] Carlisle, A., Dozier, G.: Tracking changing extrema with adaptive particle swarm optimizer. In: Proceedings of the World Automation Congress, pp. 265–270 (2002)
[22] Chaudhry, S., Luo, W.: Application of genetic algorithms in production and operations management: A review. International Journal of Production Research 43(19), 4083–4101 (2005)
[23] Cicirello, V., Smith, S.: Ant colony control for autonomous decentralized shop floor routing. In: Proceedings of the 5^{th} International Symposium on Autonomous Decentralized Systems, pp. 383–390. IEEE Computer Society Press, Los Alamitos (2001)
[24] Cobb, H.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA (1990)
[25] Cobb, H., Grefenstette, J.: Genetic algorithms for tracking changing environments. In: Proceedings of the 5^{th} International Conference on Genetic Algorithms, pp. 523–530. Morgan Kaufmann, San Francisco (1993)
[26] De Jong, K.: An analysis of the behaviour of a class of genetic adaptive systems. PhD dissertation, University of Michigan (1975)
[27] Di Caro, G., Dorigo, M.: AntNet: A mobile agents approach to adaptive routing. Tech. Rep. IRIDIA/97-12, Université Libre de Bruxelles, Belgium (1997)
[28] Di Caro, G., Dorigo, M.: An adaptive multi-agent routing algorithm inspired by ants behavior. In: Proceedings of 5^{th} Annual Australasian Conference on Parallel Real Time Systems, pp. 261–272 (1998)
[29] Di Caro, G., Dorigo, M.: Ant colonies for adaptive routing in packet-switched communications networks. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 673–682. Springer, Heidelberg (1998)
[30] Di Caro, G., Dorigo, M.: AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
[31] Di Caro, G., Dorigo, M.: Mobile agents for adaptive routing. In: Proceedings of the 31^{st} Annual Hawaii International Conference on System Sciences, pp. 74–83. IEEE Computer Society, Los Alamitos (1998)
[32] Di Caro, G., Dorigo, M.: Two ant colony algorithms for best-effort routing in datagram networks. In: Proceedings of the 10^{th} IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 541–546. IASTED/ACTA Press (1998)
[33] Di Caro, G., Ducatalle, F., Gambardella, L.: AntHocNet: An ant-based hybrid routing algorithm for mobile ad hoc networks. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 461–470. Springer, Heidelberg (2004)

[34] Di Caro, G., Ducatalle, F., Gambardella, L.: AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transaction on Telecommunications - Special Issue on Self-organisation in Mobile Networking 16, 443–455 (2005)
[35] Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: New Ideas in Optimization, pp. 11–32. McGraw-Hill, London (1999)
[36] Dreo, J., Siarry, P.: An ant colony algorithm aimed at dynamic continuous optimization. Applied Mathematics and Computation 181, 457–467 (2006)
[37] Dror, M., Powell, W.: Stochastic and dynamic models in transportation. Operations Research 41, 11–14 (1993)
[38] Ducatelle, F., Di Caro, G., Gambardella, L.: Ant agents for hybrid multipath routing in mobile ad hoc networks. In: Proceedings of Wireless On-demand Network Systems and Services, pp. 44–53 (2005)
[39] Eberhart, R., Kennedy, J.: A new optimizer using particles swarm theory. In: Proceedings of the 6^{th} International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
[40] Eyckelhof, C., Snoek, M.: Ant systems for a dynamic TSP: Ants caught in a traffic jam. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 88–99. Springer, Heidelberg (2002)
[41] Gauthier, S.: Solving the dynamic aircraft landing problem using ant colony optimisation. Masters Thesis, School of Information Technology, Bond University (2006)
[42] Godwin, T., Gopalan, R., Narendran, T.: Locomotive assignment and freight train scheduling using genetic algorithms. International Transactions in Operational Research 13(4), 299–332 (2006)
[43] Goldberg, D., Smith, R.: Nonstationary function optimization using genetic algorithm with dominance and diploidy. In: Proceedings of the 2^{nd} International Conference on Genetic Algorithms on Genetic algorithms and their application, pp. 59–68. Lawrence Erlbaum Associates, Inc., Mahwah (1987)
[44] Grefenstette, J.: Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In: Proceedings of the Congress on Evolutionary Computation, vol. 3, pp. 2031–2038. IEEE Press, Los Alamitos (1999)
[45] Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
[46] Guntsch, M., Middendorf, M.: Applying population based ACO to dynamic optimization problems. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 111–122. Springer, Heidelberg (2002)
[47] Guntsch, M., Middendorf, M.: A population based approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 72–81. Springer, Heidelberg (2002)
[48] Guntsch, M., Middendorf, M., Schmeck, H.: An ant colony optimization approach to dynamic TSP. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 860–867. Morgan Kaufmann, San Francisco (2001)

[49] Gutjahr, W., Rauner, M.: An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. Computers and Operations Research 34, 642–666 (2007)
[50] Hadad, B., Eick, C.: Supporting polyploidy in genetic algorithms using dominance vectors. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 223–234. Springer, Heidelberg (1997)
[51] Hendtlass, T.: WoSP: A multi-optima particle swarm algorithm. In: Proceedings of the Congress of Evolutionary Computing, pp. 727–734. IEEE Press, Los Alamitos (2005)
[52] Heusse, M., Snyers, D., Guérin, S., Knutz, P.: Adaptive agent-driven routing and load balancing in communication networks. Adaptive Complex Systems 2, 1–15 (1998)
[53] Holland, J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
[54] Hu, X., Eberhart, R.: Adaptive particle swarm optimisation: Detection and response to dynamic systems. In: Proceedings of the Congress on Evolutionary Computing, pp. 1666–1670. IEEE Press, Los Alamitos (2002)
[55] Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer. In: Proceedings of the Congress on Evolutionary Computing, pp. 1666–1670. IEEE Press, Los Alamitos (2003)
[56] Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer for dynamic optimization problems. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 513–524. Springer, Heidelberg (2004)
[57] Karaman, A., Uyar, S., Eryigit, G.: The memory indexing evolutionary algorithm for dynamic environments. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 563–573. Springer, Heidelberg (2005)
[58] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE Conference on Neural Networks, pp. 1942–1947 (1995)
[59] Kókai, G., Christ, T., Frühauf, H.: Using hardware-based particle swarm method for dynamic optimization of adaptive array antennas. In: Proceedings of Adaptive Hardware and Systems, pp. 51–58 (2006)
[60] Liles, W., De Jong, K.: The usefulness of tag bits in changing environments. In: Proceedings of the Congress on Evolutinary Computation, vol 3, pp. 2054–2060 (1999)
[61] Louis, S., Johnson, J.: Solving similar problems using genetic algorithms and case-based memory. In: Bäck, T. (ed.) Proceedings of the 7^{th} International Conference on Genetic Algorithms, pp. 283–290 (1997)
[62] McAllester, D., Selman, B., Kautz, H.: Evidence for invariants in local search. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 321–326 (1997)
[63] Menai, M.: An Evolutionary Local Search Method for Incremental Satisfiability. In: Buchberger, B., Campbell, J. (eds.) AISC 2004. LNCS, vol. 3249, pp. 143–156. Springer, Heidelberg (2004)

[64] Meng, Y., Kazeem, Q., Muller, J.: A hybrid ACO/PCO control algorithm for distributed swarm robots. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 273–280 (2007)
[65] Morrison, R., De Jong, K.: A test problem generator for non-stationary environments. In: Proceedings of the Congress on Evolutionary Computation, pp. 2047–2053 (1999)
[66] Moser, I.: Applying extremal optimisation to dynamic optimisation problems. PhD in information technology, Swinburne University of Technology. Faculty of Information and Communication Technologies (2008)
[67] Moser, I., Hendtlass, T.: Solving dynamic single-runway aircraft landing problems with extremal optimisation. In: Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling, pp. 206–211 (2007)
[68] Mullen, P., Monson, C., Seppi, K.: Particle swarm optimization in dynamic pricing. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1232–1239 (2006)
[69] Ng, K., Wong, K.: A new diploid scheme and dominance change mechanism for non-stationary function optimization. In: Proceedings of the 6^{th} International Conference on Genetic Algorithms, pp. 159–166. Morgan Kaufmann, San Francisco (1995)
[70] Parsopoulos, K., Vrahatis, M.: UPSO: A unified particle swarm optimization scheme. In: Proceedings of the International Conference on Computational Methods in Sciences and Engineering, pp. 868–873 (2004)
[71] Parsopoulos, K., Vrahatis, M.: Unified particle swarm optimization in dynamic environments. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 590–599. Springer, Heidelberg (2005)
[72] Pekala, M., Schuster, E.: Dynamic optimization of a heterogeneous swarm of robots. In: Proceedings of the 10^{th} IASTED International Conference on Intelligent Systems and Control, pp. 354–359 (2007)
[73] Perkins, C., Royer, E.: Ad-hoc on-demand distance vector routing. In: Proceedings of the 2^{nd} IEEE Workshop on Mobile Computing Systems and Applications, pp. 90–100 (1999)
[74] Ramsey, C., Grefenstette, J.: Case-based initialization of genetic algorithms. In: Forrest, S. (ed.) Proceedings of the 5^{th} International Conference on Genetic Algorithms, pp. 84–91 (1993)
[75] Randall, M.: A dynamic optimisation approach for ant colony optimisation using the multidimensional knapsack problem. In: Recent Advances in Artificial Life, Advances in Natural Computation, vol. 3, pp. 215–226. World Scientific, Singapore (2005)
[76] Saleh, M., Ghani, A.: Adaptive routing in packet-switched networks using agents updating methods. Malaysian Journal of Computer Science 16, 1–10 (2003)
[77] Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L.: Ant-based load balancing in telecommunications networks. Adaptive Behavior 2, 169–207 (1996)
[78] Simões, A., Costa, E.: An immune-system-based genetic algorithm to deal with dyamic environments: Diversity and memory. In: Proceedings of the 6^{th} International Conference on Artificial Neural Networks, pp. 168–174 (2003)

[79] Simões, A., Costa, E.: Improving memory-based evolutionary algorithms in changing environments. Technical Report TR2007/004, CISUC (2007)
[80] Simões, A., Costa, E.: Improving memory's usage in evolutionary algorithms for changing environments. In: Proceedings of the Congress on Evolutionary Computation, pp. 276–283. IEEE Press, Los Alamitos (2007)
[81] Simões, A., Costa, E.: Variable-size memory evolutionary algorithm to deal with dynamic environments. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 617–626. Springer, Heidelberg (2007)
[82] Simões, A., Costa, E.: VMEA: Studies of the impact of different replacing strategies in the algorithm's performance and in the population's diversity when dealing with dynamic environments. Technical Report TR2007/001, CISUC (2007)
[83] Subramanian, D., Druschel, P., Chen, J.: Ants and reinforcement learning: A case study in routing in dynamic networks. In: Proceedings of the 15^{th} International Joint Conference on Artificial Intelligence, pp. 832–838 (1997)
[84] Svenson, P.: Extremal optimization for sensor report pre-processing. In: Proceedings of Signal Processing, Sensor Fusion, and Target Recognition XIII, pp. 162–171 (2004)
[85] Tinós, R., Yang, S.: Genetic algorithms with self-organized criticality for dynamic optimisation problems. The IEEE Congress on Evolutionary Computation 3, 2816–2823 (2005)
[86] Tinós, R., Yang, S.: A self-organizing random immigrants genetic algorithm for dynamic optimization problems. Genetic Programming and Evolvable Machines 8(3), 255–286 (2007)
[87] Trojanowski, K., Michalewicz, Z.: Searching for optima in non-stationary environments. In: Proceedings of the Congress on Evolutionary Computation, vol. 3, pp. 1843–1850. IEEE Press, Los Alamitos (1999)
[88] Ursem, R.: Multinational GAs: Multimodal optimization techniques in dynamic environments. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19–26 (2000)
[89] Varela, G., Sinclair, M.: Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In: Proceedings of the Congress on Evolutionary Computation (1999)
[90] White, T., Pagurek, B., Oppacher, F.: Connection management by ants: An application of mobile agents in network management. In: Proceedings of Combinatorial Optimization (1998)
[91] Xia, Y., Chen, J., Meng, X.: On the dynamic ant colony algorithm optimization based on multi-pheromones. In: Proceedings of the 7^{th} IEEE/ACIS International Conference on Computer and Information Science, pp. 630–635 (2008)
[92] Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1115–1122. ACM, New York (2005)
[93] Yang, S.: Associative memory scheme for genetic algorithms in dynamic environments. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 788–799. Springer, Heidelberg (2006)

[94] Yang, S.: A comparative study of immune system based genetic algorithms in dynamic environments. In: Proceedings of the 8^{th} Conference on Genetic and Evolutionary Computation, pp. 1377–1384. ACM, New York (2006)
[95] Yang, S.: Genetic algorithms with elitism-based immigrants for changing optimization problems. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 627–636. Springer, Heidelberg (2007)
[96] Yang, S., Tinós, R.: A hybrid immigrants scheme for genetic algorithms in dynamic environments. International Journal of Automation and Computing 4, 243–254 (2007)
[97] Zhang, X., Li, X., Li, Y., Zhou, Y., Zhang, J., Zhang, N., Wu, B., Yuan, T., Chen, L., Zhang, H., Yao, M., Yang, B.: Two-stage adaptive PMD compensation in 40 Gb/s OTDM optical communication system using PSO algorithm. Optical and Quantum Electronics 36, 1089–1104 (2004)
[98] Zhang, Y., Kuhn, L., Fromherz, M.: Improvements on ant routing for sensor networks. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 154–165. Springer, Heidelberg (2004)