A survey of swarm intelligence for dynamic optimization: Algorithms and applications

Michalis Mavrovouniotis[a], Changhe Li[b,*], Shengxiang Yang[c]

[a] *School of Science and Technology, Nottingham Trent University, Nottingham, NG11 8NS, United Kingdom*
[b] *School of Automation and Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems China University of Geosciences, Wuhan 430074, China*
[c] *Centre for Computational Intelligence (CCI), School of Computer Science and Informatics De Montfort University, The Gateway, Leicester LE1 9BH, United Kingdom*

Abstract

Swarm intelligence (SI) algorithms, including ant colony optimization, particle swarm optimization, bee-inspired algorithms, bacterial foraging optimization, firefly algorithms, fish swarm optimization and many more, have been proven to be good methods to address difficult optimization problems under stationary environments. Most SI algorithms have been developed to address stationary optimization problems and hence, they can converge on the (near-) optimum solution efficiently. However, many real-world problems have a dynamic environment that changes over time. For such dynamic optimization problems (DOPs), it is difficult for a conventional SI algorithm to track the changing optimum once the algorithm has converged on a solution. In the last two decades, there has been a growing interest of addressing DOPs using SI algorithms due to their adaptation capabilities. This paper presents a broad review on SI dynamic optimization (SIDO) focused on several classes of problems, such as discrete, continuous, constrained, multi-objective and classification problems, and real-world applications. In addition, this paper focuses on the enhancement strategies integrated in SI algorithms to address dynamic changes, the performance measurements and benchmark generators used in SIDO. Finally, some considerations about future directions in the subject are given.

Keywords: Swarm intelligence, Dynamic optimization, Ant colony optimization, Particle swarm optimization

1. Introduction

Swarm intelligence (SI) is an important category of optimization methods. SI is the property of a system whereby the collective behaviours of agents that interact locally with their environment cause coherent functional global patterns to emerge. Different from evolutionary algorithms (EAs), SI algorithms are inspired from simple behaviours and self-organizing interaction among agents, such as ant colonies foraging, bird flocking, animal herding, bacterial growth, honey bees, fish schooling, and so on. The term SI was first used by Beni [1] in cellular robotic system where simple agents organize themselves through neighbourhood interactions and later on established in [2, 3, 4].

The mainstream SI algorithms are ant colony optimization (ACO) [5] and particle swarm optimization (PSO) [6]. Less popular SI algorithms include artificial bee colony (ABC) [7], bacterial foraging optimization (BFO) [8], firefly algorithm (FA) [9, 10], artificial fish swarm optimization (AFSO) [11] and many others. Originally, SI algorithms were designed for stationary optimization problems. However, many real-world optimization problems are subject to dynamic environments. Changes in a dynamic optimization problem (DOP) may occur in the objective function, constraints, problem instance, Pareto front or set (in the case of dynamic multi-objective optimization problems) that cause the optimum to change. Hence, DOPs are more challenging to address than stationary optimization problems since repeated optimization of the changing optimum is required [12].

The field of dynamic optimization is closely related with EAs, known as evolutionary dynamic optimization (EDO) [12]. However, it has been a growing interest to apply SI algorithms on different DOPs. EDO has received extensive attention with several surveys [13, 12, 14, 15] and books [16, 17, 18, 19, 20], whereas SI dynamic optimization (SIDO) has not received much attention, with exception of some very brief reviews of PSO in [14] and ACO in [15] included as subsections in the EDO surveys. The aim of this paper is to extend these reviews of ACO and PSO and provide a comprehensive survey of existing work done related to SIDO, which also includes the less popular and recent SI algorithms. The survey will mainly focus on classifying SI algorithms based on their applications and reviewing the strategies integrated with SI algorithms to tackle dynamic changes. The DOPs are mainly classified into problems with discrete and continuous spaces and their applications are further classified. A review of real-world problems addressed with SI and reviews of performance measurements and benchmark generators of SIDO are also given.

The rest of the paper is organized as follows. Section 2 briefly presents the concept of DOPs and describes the differences between discrete and continuous DOPs and their appli-

*Corresponding author
Email addresses: michalis.mavrovouniotis@ntu.ac.uk (Michalis Mavrovouniotis), changhe.lw@gmail.com (Changhe Li), syang@dmu.ac.uk (Shengxiang Yang)

cations. Moreover, it describes the benchmark generators and performance measurements commonly used in SIDO. Section 3 briefly describes different SI algorithms. Section 4 reviews algorithms and applications of SIDO arranged by classes of problems, i.e., discrete, continuous, constrained, multi-objective, and classification problems. Section 5 reviews the real-world applications in which SI algorithms are used. Section 6 concludes this paper and summarizes future research issues and directions on SIDO.

2. Dynamic optimization

2.1. Dynamic optimization problem (DOP)

A DOP can be intuitively defined as a sequence of static problem instances that need to be optimized [21]. The two main aspects of "dynamism" are defined by the frequency and magnitude of an environmental change. The former and latter parameters correspond to the speed and degree at which the environment of the problem changes, respectively. Other aspects include the predictability, detectability, and time-linkage of dynamic changes [22, 23]. The former two aspects correspond to whether a dynamic change can be predicted or detected during the execution or not, respectively, and the latter corresponds to whether a decision made now to address a dynamic change is dependent on any earlier decisions or not.

An environmental change may involve the objective function (or functions if a dynamic multi-objective problem is considered [24, 25]), input variables, problem instances and constraints (e.g., dynamic constrained optimization [26]). Formally, a DOP can be defined as follows:

$$\text{DOP} = \text{optimize } f(\mathbf{x}, t) \text{ subject to } X(t) \subseteq S, t \in T, \quad (1)$$

where S is the search space, t is the time, $f : S \times T \rightarrow R$ is the objective function that assigns a value (i.e., R) to each possible solution $\mathbf{x} \in S$ and $X(t)$ is the set of feasible solutions $\mathbf{x} \in X(t) \subseteq S$ at time t [13, 15]. Each feasible solution $\mathbf{x}$ consists of optimization variables $\mathbf{x} = \{x_1, \ldots, x_n\}$. Every solution $\mathbf{x} \in X(t)$ has a a set of neighbours $\mathcal{N}(\mathbf{x}) \subseteq X(t)$ where $\mathcal{N}(\mathbf{x})$ is a function that assigns a neighbourhood to $\mathbf{x}$. A local optimum solution is a feasible solution $\mathbf{x}'$ that $f(\mathbf{x}', t) \leq f(\mathbf{x}, t), \forall \mathbf{x} \in \mathcal{N}(\mathbf{x})$ for minimization problems or $f(\mathbf{x}', t) \geq f(\mathbf{x}, t), \forall \mathbf{x} \in \mathcal{N}(\mathbf{x})$ for maximization problems, respectively.

The global optimum is a feasible solution $\mathbf{x}^*$ that minimizes (or maximizes) the objective function $f(\mathbf{x}^*, t) \leq f(\mathbf{x}, t)\ \forall \mathbf{x} \in X(t)$ for minimization problems (or $f(\mathbf{x}^*, t) \geq f(\mathbf{x}, t)\ \forall \mathbf{x} \in X(t)$ for maximization problems).

2.2. Discrete vs continuous space

There are different classes of optimization problems that differ in the definition of the search space $X(t)$. In this paper two fundamental types of problems are considered as follows:

- Discrete optimization problems where all optimization variables x_i are discrete that take values $x_i \in \mathbf{D}_i = \{v_i^1, \ldots, v_i^{|\mathbf{D}_i|}\}, i = 1, \ldots, n$.
- Continuous optimization problems where all optimization variables x_i are continuous that take values $x_i \in \mathbf{D}_i \subseteq \mathbb{R}, i = 1, \ldots, n$.

The main difference between discrete and continuous optimization problems lies in that discrete optimization problems have a finite search space. More precisely, they are characterized by a finite set of variable values, e.g., binary that are restricted to the values 0 and 1 or objects drawn from a finite set of elements. Differently, each variable value within continuous optimization problems may assume an infinite number of values, e.g., real numbers. Since computers are digital in nature, representing discrete variables is straight forward. In contrast, representing continuous variables requires to impose certain limitation since it is not possible to represent an infinite number of values.

In the context of the No-Free-Lunch (NFL)[1] theorem [27], any optimization problem that runs in a computer system consists of a finite domain, and thus, can be considered as discrete. A different approach to NFL showed that the theorem also holds for arbitrary domains and co-domains [28]. In contrast, other studies [29, 30] showed that the NFL theorem does not hold in continuous domains although its soft form of NFL holds in countably infinite domains. However, in [31], it was proved that NFL does hold in continuous domains because the conclusions of the authors in [29, 30] are drawn from their imposition of the artificial constraint that functions under consideration be measurable. Additional artificial constraints are imposed but measurability was enough to trivialize NFL.

Nevertheless, the way of addressing discrete and continuous problems typically differs. For example, in the case of discrete problems, the set of available values are predefined before starting the optimization process. Hence, an appropriate solver must select the best possible combination of values to find a solution. Such approach may not be efficient in the case of continuous problems. Instead, a flexible floating point representation of real-valued variables is typically used by the solver. In this way, a more efficient way to find a solution with the required accuracy is allowed.

2.3. Applications

Both discrete and continuous optimization problems have a wide range of applications to be summarized in Tables 2 and 3, respectively, later. Most practical real-world problems in the fields of transportation, scheduling, management, production, facility control, consist of a finite number of possible solutions. Therefore, they can be formulated as discrete optimization problems.

For example, many real-world optimization problems with network environments, such as road systems, social networks, telecommunication networks, rail networks, and so on, are often modelled by weighted graphs. A fundamental discrete optimization problem that is modelled by a weighted graph is the

[1] NFL theorem roughly states that all search heuristics (like the SI solvers discussed in this paper) have the same performance when averaged over all possible functions in finite search domains.

travelling salesman problem (TSP). The TSP has many applications in both routing and scheduling problems, such as the vehicle routing problem (VRP), which is closely related in the field of transportation, distribution of goods and logistics. The arc routing counterpart of the VRP, i.e., the capacitated arc routing problem, has also many applications in the real world, such as the salt routing optimization problem, urban waste collection problem and snow removal problem.

Differently, the applications in continuous optimization include practical problems in computational finance, the training of an artificial neural network that may be used for medical diagnoses, prediction of traffic in a road system, voice and face recognition, and forecasting weather or customer demands. Moreover, it may include the design of optimal shapes such as wings, turbines, engines, power plants and others.

2.4. Measurements

The aim in tracking the moving optimum (TMO) is to find the best solution of the environment at any time. Researchers view their algorithms from different perspectives in TMO [32]. Some researchers pay more attention on extreme behaviours of a system, in particular, the best that the system can do, e.g., modified offline performance [33, 17], collective mean fitness [34], best before change [35, 36]. Differently, other researchers want to observe "how close to the moving optimum a solution found by an algorithm is" [37, 38]. Therefore, the measurements require that the global optimum value is known during the dynamic changes, e.g., the offline error [39], average score [40], accuracy [41] and other measurements based on the distance between the solutions found by the algorithm and the global optimum [42, 43]. Others are concerned for measures which can characterize the population as a whole, e.g., the average performance or the average robustness [44].

Recently, a new perspective on DOPs has been established, known as robust optimization over time (ROOT), where the target is to find the sequence of solutions which are robust over time [45]. Particularly, a solution is robust over time when its quality is acceptable to the environmental changes during a given time interval. Regarding SI algorithms, so far, TMO has been mainly used with all of them. In contrast, ROOT has been used only with PSO algorithms [46]. Robust optimization was found to be useful when dealing with problem uncertainties with PSO [47] and ACO [48, 49], e.g., reducing computational efforts.

Apart from measurements that involve the performance of algorithms, other measurements involve the behaviour of algorithms. Popular examples are: the diversity of solutions [50, 51, 52, 39], stability [53, 41], reactivity [41], robustness [44], cross-entropy [54], peak cover [17] and λ-branching [55][2]. Other performance measurements were proposed exclusively to evaluate the ability of the algorithms to track and locate feasible regions such as: feasible time ratio, optimal region tracking measure, local search cover, number of evaluations for constraints. In addition, existing measurements were modified for dynamic constrained optimization problems such as: the peak cover to count the number of only feasible regions in each period and the offline error to consider the best error as normal but if there is no feasible solution; then the current worst possible value is considered [56].

Of course, all the above measurements for DOPs assume a single objective. Different measurements are used when dealing with multiple objectives such as: spacing [57], hypervolume ratio [58], S- and FS-metrics [59], accuracy [60], stability [60], variable space generational distance [61] and maximum spread [61].

2.5. Benchmark generators

Benchmark generators are essential tools in DOPs, due to the limited theoretical work available in SIDO and generally in dynamic optimization [62, 63]. They enable researchers to develop and evaluate new algorithms for DOPs, and more importantly to compare them with existing ones.

2.5.1. The generation of dynamics

A straightforward method, but not efficient, to construct a dynamic test problem is to switch between different static instances that will cause an environmental change [64]. The benchmark problems that generate dynamic environments following this methodology are specified for a single problem. In some other cases, researchers prefer to create their own customized benchmark problems that aim to model some real-world scenarios [65, 66, 37, 67, 68, 69] which again are developed for a specific problem, or even a specific instance of the problem.

Several general purpose dynamic benchmark generators have been proposed that re-shape the fitness landscape (for continuous problems) or move the search process to a different location of the fitness landscape (for discrete problems). Comprehensive surveys can be found in [13, 14]. Probably the most commonly used benchmark generators for DOPs are: (1) the moving peaks benchmark (MPB) [4]; (2) the generalized dynamic benchmark generator (GDBG) [40]; (3) the exclusive-or (XOR) DOP generator for binary-encoded problems [70]; and (4) the dynamic benchmark generator for permutation-encoded problems (DBGP) [71]. The first two benchmark generators work for the continuous domain where they use functions with adjustable parameters to simulate shifting landscapes. Considering that the continuous space can be modelled as a "field of cones" [72], then each cone can be adjusted individually to represent different dynamics. Fig. 1 shows the fitness landscapes of two cases that belong to the MPB and the GDBG, respectively.

Since the continuous space has an infinite number of variable values, certain limitations are imposed to develop benchmarks to solve complex mathematical functions. The MPB [4] is one of the mostly used benchmarks for testing the performance of algorithms in the continuous space. Each peak in the MPB problem is a cone shape. This would be easy for an algorithm to exploit a local optimum in the fitness landscape. To overcome the limitation of DOPs like the MPB and DF1 [72] (a similar

[2]Cross-entropy and λ-branching are especially designed for ACO to gather information of the distribution of the pheromone trails.

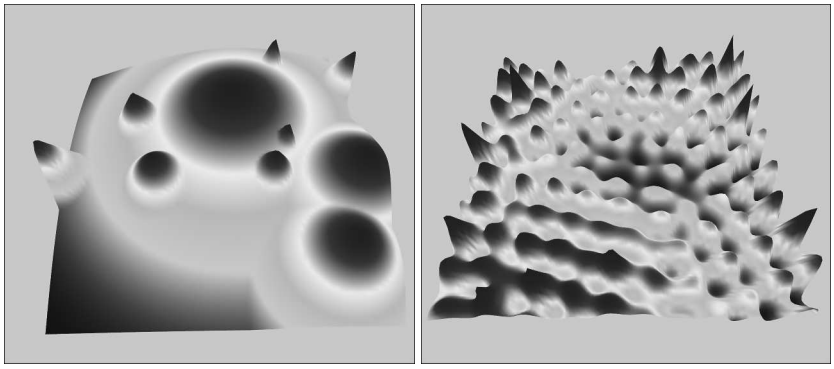

Fig. 1: The landscape of the MPB (left) and the GDBG (right).

benchmark generator with MPB), the GDBG benchmark was developed by Li et al. [40], which was initially proposed for the 2009 IEEE Competition on Evolutionary Computation for DOPs. The GDBG has a more complex fitness landscape than the MPB problem due to a huge number of rotated optima in the search space. In the GDBG benchmark, there are six basic test functions and each test function has eight change types, which are the small step change, large step change, random change, chaotic change, recurrent change, recurrent change with noise, dimensional change and number of peaks change. In [45], the MPB was modified to generate ROOT problems. In the original MPB for TMO, all the peaks are changed at the same frequency and severity, whereas on the modified MPB for ROOT each peak has its own frequency and severity.

In discrete spaces, the landscape is indistinct and cannot be defined without reference to the optimization algorithm [63]. Usually, the components that define the discrete/combinatorial optimization problem are modified and are specific to the problem. For the dynamic TSP (DTSP), when a change occurs, either nodes may be replaced/removed [66] or the weights of the arc may increase/decrease [65, 21]. The dynamic changes in dynamic VRP (DVRP) may occur on the weights of the arcs [33] or new customers may be revealed [67]. Dynamic changes occur when new jobs arrive during the execution for the dynamic job shop scheduling problem (DJSSP) [73, 74]. The changes in the dynamic knapsack problem (DKP) may occur on the value and weight of items as well as the capacity of knapsack or directly to the objective function [75]. The XOR DOP can generate a DOP for any binary-encoded problem, which are special case of discrete problems, by flipping the values from "0" to "1" and vice verse, according to binary templates. The DBGP can generate a DOP for any permutation-encoded routing problem, by swapping the nodes of the problem instance [55, 76].

The MPB, GDBG, DBGP, XOR DOP benchmark generators and all other aforementioned benchmarks assume unconstrained and single objective optimization problems. Recently, benchmark generators for continuous dynamic constrained optimization [77, 78, 26, 14] and continuous dynamic multi-objective optimization [25, 61, 79, 80, 81, 82, 83, 84, 85] are proposed. But, constrained and multi-objective optimization under the discrete space has not attracted much attention yet and deserves future consideration.

2.5.2. Discussion

For benchmarks in the continuous space, most of them take the following form to generate landscape: $f(x,t) = max g_i(x,t)$. This way is simple and straightforward. However, it has two disadvantages. Firstly, it is computationally inefficient. To evaluate a solution, we have to compute the objective value of every g, and then find the best objective value as the objective value of the solution. Secondly, some peaks may be invisible if they are covered by higher peaks. In addition, these benchmarks lack of scenarios of real-world problems.

For benchmarks in the discrete space, the variables of the problem instance are typically modified, e.g., the nodes or arcs of a weighted graph $G = (C, L, t)$ that represent the problem. Although real-world scenarios are generated with most benchmarks the global optimum value is unknown during dynamic changes. Hence, if all compared algorithms are performing far away from the optimum, one will not be able to observe it. A solution to that is to move the search process into a different location of the landscape rather than changing it, e.g., using the DBGP (for permutation-encoded problems) and the XOR DOP (for binary-encoded problems) that can generate dynamic environments without affecting the global optimum value. Basically, the encoding of the problem instance is modified rather than the search space. However, the real-world scenario ability is sacrificed for the sake of benchmarking.

3. Swarm intelligence algorithms

3.1. Brief description

The area of metaheuristics in SI is chaotic because there are many "novel" metaheuristics that are basically repeating existing metaheuristics [87] or are not even inspired by nature or swarms, e.g., the fireworks algorithms inspired by the fireworks explosions [88]. Nevertheless, this paper focuses only on the SI metaheuristics applied to DOPs as listed in Table 1.

Generally speaking, all SI algorithms were developed specifically for different optimization problem domains as defined in Table 1. For example, ACO was developed for a discrete space whereas the remaining algorithms in Table 1 for a continuous space. The common characteristics of these algorithms

Table 1: Swarm intelligence algorithms applied on DOPs so far

Algorithm	Main Reference	Initial Problem Domain	Exploitation Mechanism	Exploration Mechanism
Ant colony optimization (ACO)	[5]	Discrete optimization	Construction of solutions according to heuristic information	Consideration of pheromone trail values
Particle swarm optimization (PSO)	[86]	Continuous optimization	Update the particle positions towards the global best particle	Velocity update of particles
Artificial fish swarm optimization (AFSO)	[11]	Continuous optimization	Follow and swarm behaviour	Prey behaviour
Bacterial foraging optimization (BFO)	[8]	Continuous optimization	Chemotaxis and reproduction steps	Elimination-dispersal step
Artificial bee colony (ABC)	[7]	Continuous optimization	Neighbourhood search carried by employed and onlooker bees	Random search of scout bees
Firefly algorithm (FA)	[9]	Continuous optimization	Firefly movement according to attractiveness	Random move of the best firefly

are that they are inspired from nature, population-based, and iterative. Their differences, apart from their behaviour inspiration, lie in the way the search space is explored and exploited by the "agents" [89].

3.1.1. Ant colony optimization (ACO)

ACO was inspired by the foraging behaviour of real ants. The goal of ants is to find the shortest path between their nest and food sources. ACO metaheuristic is based on several construction steps and on a dynamic memory structure that contains information regarding the quality of previously obtained results [90, 91]. Each ant represents a potential solution of the problem. ACO consists of a forward mode where ants construct their solutions probabilistically based on existing pheromone trails and heuristic information available a priori. When all ants complete their forward mode they switch to their backward mode where a shared pheromone table is updated accordingly, i.e., the better the solution quality the more pheromone deposited.

There are two main ACO frameworks, i.e., evaporation-based [92, 93] and population-based [66]. Their difference lies in the way pheromone is updated. The evaporation-based framework reduces the pheromone trails gradually by a constant amount to eliminate any previous poor old "decisions". The population-based framework uses a population that removes pheromone trails directly when a solution is removed from the population.

3.1.2. Particle swarm optimization (PSO)

PSO was first introduced in [86] to address continuous optimization problems. Each particle represents a potential solution of the problem. More precisely, each particle consists of a velocity and position vectors, respectively, which are updated according to the best so far position of the particle and the best so far position of the swarm.

There are two main models of the PSO algorithm, i.e., the global best and local best, respectively. Their difference lies in the neighbourhood structure for each particle. In the global best model, the neighbourhood of a particle consists of the particles in the whole swarm, which share information between each other. On the contrary, in the local best model, the neighbourhood of a particle is defined by several fixed particles. Poli et al. [94] stated that the global best model converges faster than the local best model whereas the former model has a higher probability of getting stuck in local optima than the latter model. Surveys of different PSO variations can be found in [95, 96].

Both models are used but in different ways due to their characteristics. The global best model is normally used in multi-swarm based algorithms [97, 98, 99], while the local best model is commonly used in algorithms with a single swarm [100, 101, 102].

3.1.3. Artificial bee colony (ABC)

There are several developments of bee-inspired algorithms such as: ABC, bee colony optimization, bee system, marriage process bee, honey bee mating optimization, virtual bee algorithm, honey bee algorithm and beehive algorithm. Surveys of the different developments can be found in [103, 104, 105]. In this paper, we mainly focus on the ABC algorithms that have attracted most of the attention, especially in DOPs [105]. In particular, an ABC algorithm mimics the behaviour of real bees colonies [7]. A conventional ABC algorithm consists of food sources, whereas each food source represents a potential solution of the problem. Food sources are updated by three groups of bees: employed, onlooker and scout bees.

Within the employed bee phase, bees search for new solutions. In particular, each bee produces a new candidate food source position from the old one. In the case that food sources with more nectar are found, i.e., the new solutions have better fitness than the current, then they are updated. Next, the relative probabilities according to the fitness determined from the employed bee phase are determined in the onlooker bee phase. Then, onlooker bees select a solution probabilistically in which the fittest solutions have a higher probability to be selected by onlooker bees. After that, onlooker bees have the same be-

haviour with the employed bees. Finally, scout bees randomly reallocate solutions if they are abandoned, e.g., they have not been updated for a certain time.

3.1.4. Bacterial foraging optimization (BFO)

The BFO algorithm was inspired by the complex organized activities in bacterial foraging and the survival of bacteria in different environments [8, 106, 107]. A BFO algorithm consists of several bacteria, which represent solutions in the optimization problem and consists of three processes: chemotaxis, reproduction, and elimination-dispersal.

In chemotaxis, a bacterium with random direction represents a tumble and a bacterium with the same direction of the previous step indicates a run. Next in the reproduction process all bacteria is sorted and only half of the fittest bacteria survive. Then, the surviving bacteria is split into two identical ones to form the new bacteria. Finally, in the elimination-dispersion process, a bacterium is chosen probabilistically to move to a different random position in the search space. Although this action maintains the diversity during execution, it may disturb the optimization process and therefore it is performed after several steps of the reproduction process.

3.1.5. Artificial fish swarm optimization (AFSO)

There are several existing developments of fish-inspired algorithms. A detailed description of developments can be found in [108]. In this paper, we focus on the AFSO inspired by the foraging behaviour of real fish swarms in water world [11] which was applied for DOPs. Within AFSO, each artificial fish looks for a position (solution) with more food source (better fitness) by performing three main behaviours: prey, swarm and follow. The prey behaviour is performed by an artificial fish without considering other swarm members. More precisely, a target position better than the current is considered randomly within the visual of the fish.

The swarm behaviour is a group behaviour and is performed globally among all members of swarm as follows. Each artificial fish consists of a number of neighbours within its visual. If the central position of the visual field is better; then it moves towards the central position; otherwise, the prey behaviour is performed again. Similarly, the follow behaviour is performed, but instead of moving toward the central position, the artificial fish will move toward a better neighbour position within its visual. Otherwise, the prey behaviour is performed again. Basically, the prey behaviour is performed when an artificial fish is not able to move to a better position when the follow or swarm behaviour is performed. If the algorithm reaches stagnation behaviour some artificial fishes are selected randomly from the whole artificial fish swarm and are set randomly. The best so far artificial fish position (i.e., solution) is recorded.

3.1.6. Firefly algorithm (FA)

The FA was inspired by the flashing patterns and behaviour of fireflies [9, 10]. An FA is based on three assumptions:

1. all fireflies can be attracted by all other fireflies
2. the attractiveness of each firefly is proportional to the brightness of other fireflies
3. the landscape of the problem determines the brightness of fireflies.

Hence, a firefly that is less bright will move toward a more bright one. Otherwise, if a firefly is not able to locate a brighter firefly, it will move randomly. Each firefly glows proportionally to its solution quality, which, together with its attractiveness, dictates how strong it attracts other members of the swarm.

3.2. Adapting in changing environments

Since all SI algorithms were initially designed for stationary optimization problems, they share a common property: convergence, i.e., they are designed to converge to the optimum quickly and precisely. In contrast, DOPs require repeated optimization and tracking of the moving optimum. However, when an algorithm converges, its adaptation capabilities are lost due to the diversity loss problem. Therefore, it is important to address the diversity loss problem by increasing/maintaining the diversity. However, it does not mean that a high level of diversity will lead to better performance [12, 55]. This is because too much randomization may disturb the optimization process.

Another important aspect of SI algorithms to adapt well in DOPs is to promote the knowledge transfer. Naturally, knowledge can be transferred from previously optimized environments using SI algorithms, e.g., via pheromone trails with ACO, via the food sources with ABC and AFSO, via the position of fireflies and bacteria with FA and BFO, respectively. However, it may not be enough to quickly recover when a dynamic change occurs. On the other hand, if too much knowledge is transferred, it may start the optimization process close to a poor local optimum and get stuck there.

Enhanced SI algorithms have proved to be powerful for different DOPs. The main idea of enhancement strategies integrated to SI algorithms is to achieve a good balance for *knowledge transfer* and *diversity maintenance* as shown in Fig. 2. In addition, these two factors are also conflicting because if diversity is not maintained then the algorithm will not be very flexible to utilise any knowledge transferred.

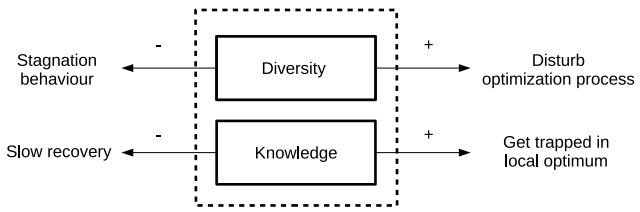

Fig. 2: Effects of tuning diversity maintenance and knowledge transfer of SI algorithms in DOPs

4. Swarm intelligence for dynamic optimization

In this section, major SI strategies that tackle dynamic changes will be reviewed for various classes of problems di-

vided as follows:

- SI in dynamic discrete optimization
- SI in dynamic continuous optimization
- SI in dynamic constrained optimization
- SI in dynamic multi-objective optimization
- SI in dynamic classification

Since the first two classes have been extensively researched, they are further classified by application (see Tables 2 and 3) and by the type of strategy: (a) increasing diversity after a change, (b) maintaining diversity during execution, (c) memory schemes, (d) multiple population[3] methods and (e) hybridizations. The remaining classes are current trends in SIDO and their research is limited.

4.1. SI in dynamic discrete optimization

Many discrete optimization problems, either in stationary or dynamic environments, are $\mathcal{NP}$-hard, i.e., it is strongly believed that they cannot be solved to optimality within polynomial computation time [109]. Dynamic versions of several popular discrete optimization problems have been addressed using SI methods.

From Table 2, it can be observed that ACO is mostly used for discrete/combinatorial problems. Typically, these problems are represented using weighted graphs $G = (C, L)$, where C is a set of components and L is a set of links. For example, for the DTSP or DVRP, the pheromone trails τ_{ij} are mapped with both the components and links of the problem to represent the desirability of visiting component (node) j after component (node) i. Similarly, the pheromone trails in DJSSP refer to the desirability of choosing operation j directly after operation i. For binary-encoded problems, pheromone trails are associated to the two possible choices that a component can take. For the DKP the pheromone trails τ_i are associated only with the components and refer to the desirability of adding item i. Therefore, when a dynamic change occurs, some of the current pheromone trails will not be compatible with the characteristics of the new environment whereas some others will contain information useful to guide optimization into promising areas of the search space quickly.

Several strategies were proposed to address this issue categorized in the following subsections.

4.1.1. Increasing diversity after a change

Many strategies in DOPs depend on the detection of dynamic changes. The detection is typically performed by re-evaluating a solution and whenever a change in its fitness occurs then a dynamic change is detected. The strategies in this category performs actions whenever a change is detected to increase diversity.

1. Complete restart. The pheromone trails of ACO algorithms are re-initialized with an equal amount whenever a change is detected for DTSP and DVRP [44, 55, 65]. This corresponds to a complete restart of the algorithm that optimizes each dynamic change from scratch. Since the initial phase of an algorithm is highly explorative, the diversity is increased significantly. However, all previous knowledge gained is destroyed.
2. Partial restart. Other strategies aim to simply increase the diversity and maintain the knowledge gained, simultaneously, when a change occurs. For example, Guntsch and Middendorf [54, 111] proposed partial restart strategies using local information, e.g., the η-strategy and τ-strategy, which take into account where the dynamic change actually occurs, e.g., which cities are added/removed for DTSP. The aim of both strategies is to give a higher degree of re-initialization to the pheromone trails closer to the offended areas. This corresponds to a partial restart of the algorithm because knowledge is maintained and could speed up the optimization process. However, apart from detecting the period of change, the location of changes needs to be detected. It may not be always available or requires extensive computational efforts to detect them. A similar partial restart strategy via pheromone trails was proposed for the DJSSP [136, 73].
   Angus and Hendtlass [110] used the old pheromone trails and modified proportionally to the maximum pheromone trail value for DTSP. Similarly, Eyckelhof and Snoek [65] proposed a "shaking technique" to regulate the previous pheromone trails. All pheromone trails are modified using a logarithmic formula, where the pheromone values closer to the initial pheromone value are deducted slightly, whereas higher pheromone values are deducted severely. The same shaking technique was adopted in a PSO variation that uses a pheromone table as a communication topology for particles [118, 119]. However, the results were not as promising as with ACO algorithms that are more effective for graph problems.
   The above strategies performed modification via the pheromone trails directly. In contrast, Rand and Riolo [75] repaired the best solution of the previous environment for the DKP, since it became infeasible by the change, using a deconstruction technique. The pheromone trails are affected indirectly according to the changes made to the solutions during the repair. In this way, partial knowledge of the previous environment is preserved. On the same problem but with a different type of dynamic changes that does not affect the representation of the problem, Baykasoğlu and Ozsoydan [137] enhanced the diversity of FA by partially restarting a part of the population when a dynamic change occurs.
   Montemanni et al. [67] addressed the DVRP by dividing the working day into time slices. Basically, the dynamic problem is split into a series of several static problems that are optimized by ACO separately. Pheromone conservation of the previous static problem was used to prevent starting the optimization from scratch for the next

[3] Population is used as a general term to define a set of agents. For different SI algorithms, a population is defined differently, e.g., as *colony* for ACO and ABC, as *swarm* for PSO, AFSO and FA, and as *bacteria* for BFO.

Table 2: Swarm intelligence applications for discrete DOPs

ACO	PSO	ABC	BFO	AFSO	FA
		Dynamic Travelling Salesman Problem			
[110, 54, 111, 66, 44, 65, 55, 21, 112, 113, 114, 115, 35, 55, 116, 36, 117]	[118, 119, 120, 121]	–	–	–	–
		Dynamic Vehicle Routing Problem			
[67, 122, 123, 124, 125, 48, 49, 37, 33, 126, 127]	[128, 129, 130, 131, 132, 133, 134, 47, 135]	–	–	–	–
		Dynamic Job Shop Scheduling Problem			
[136, 73, 74]	–	–	–	–	
		Dynamic Knapsack Problem			
[75]	–	–	–	–	[137]
		Other Binary-Encoded Functions			
[138, 76]	–	–	–	–	–

Table 3: Swarm intelligence applications for continuous DOPs

ACO	PSO	ABC	BFO	AFSO	FA
		Moving Peaks Benchmark			
-	[140, 141, 142, 143, 144, 145, 146, 147, 148, 98, 149, 99, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 97, 164, 102, 165, 166, 101, 167, 168]	[169, 170, 171, 172]	[173, 174, 175]	[176, 177]	[178, 179, 180, 181, 182, 183]
		Generalize Dynamic Benchmark Generator			
[184, 185]	[186, 187]	[188, 189]	–	–	–
		Other Time-Varying Functions			
[190, 191]	[192, 193, 194]	[195, 196, 197]	–	–	–

problem. Using time slices, the detection of change is not necessary. However, the performance of the algorithm depends on the number of time slices that is predefined. A similar ACO approach was proposed in [122] and a PSO approach in [135]. More PSO variations using time slices were proposed by Khouadjia et al. [128, 129]. Each particle keeps track of its best solution which is later used to form the initial swarm for the next time slice. In contrast, the initial swarm of the next time slice in [133] is defined by the previous global best within a given radius.

4.1.2. Maintaining diversity during execution

There are strategies that do not necessarily require the detection of change to increase diversity. Diversity is maintained during the execution. In fact, unmodified ACO algorithms were applied to the DJSSP [74] and binary-encoded problems [138, 76]. Because pheromone evaporation helps to forget unused pheromone trails, it helps the adaptation process when a dynamic change occurs. Another way is to gather statistics from the pheromone trails to detect stagnation and re-initialize the pheromone trails [139]. This way was used to the aforementioned PSO with pheromone matrix for the DTSP [121].

Based on the pheromone evaporation effect, the pheromone evaporation rate ρ was adapted during the optimization process [55] for DTSP and DVRP to speed up the adaptation. The idea is to increase the evaporation rate when the algorithm is approaching stagnation behaviour to eliminate pheromone trails faster. Since the adaptive method required a fixed step size; a self-adaptive evaporation rate was later proposed to address this issue [116]. In particular, the evaporation rate is discretized to

a set of values with which pheromone trails are associated. The ants are responsible to select the appropriate rate at each iteration.

Ankerl and Hämmerle [125] self-adapted the α and β parameters that control the weight of pheromone trails and heuristic information, respectively. Each ant uses a different set of these parameters, which initially are the same. This way enables ants to explore different locations of the search space, and thus, maintain diversity. Liu [117] modified the decision rule of ACO instead of adapting the control parameters. A rank-based non-linear selection pressure function is integrated to the decision rule. Although the integration enhances diversity, it causes the introduction of new parameters that need to be optimized.

A very popular strategy integrated with ACO is the immigrants scheme [55, 21, 112, 113, 33, 127]. The general idea is to introduce immigrant ants, either randomly or using the best from the previous environment, to deposit pheromone trails. In this way, diversity is maintained. However, if too many random immigrant ants are generated, they may disturb the optimization process.

4.1.3. Memory schemes

Another way to transfer knowledge from previous environments is to use an external memory. Typically, promising solutions are stored in the memory and reused when a dynamic change occurs. Again, this type of strategies require the detection of change to update the memory or even repair it.

One of the most popular ACO for the DTSP is the population-based ACO (P-ACO) [66]. P-ACO maintains a memory of the best ants that is used for updating the pheromones. When a dynamic change occurs, the ants in the memory are repaired heuristically, and thus, the pheromone values are modified. Similarly, with the η-strategy and τ-strategy mentioned above, the location of changes is required for the heuristic repair.

The memory-based immigrants ACO [37, 21, 33] uses a limited size memory to store the best representative solution from several previously optimized environments. The idea is to generate immigrant ants based on the currently best memory to transfer knowledge and increase diversity simultaneously.

4.1.4. Multiple population methods

The use of separate populations can naturally increase and maintain diversity. The idea is to allocate different populations into different areas in the search space.

A multi-colony ACO algorithm where each colony uses a separate pheromone table in an attempt to maximize the search area explored was proposed for the DTSP [36] and DVRP [123, 124]. An explicit method to keep the colonies into different areas was not applied, but the results showed that it performs better than a single colony ACO algorithm. In contrast, multi-colony ACO in which different colonies, called castes, use the same pheromone table was proposed in [114, 115]. Each caste uses different parameter settings that correspond to different behaviour for each caste that can cover different location of the search space. A similar idea was adopted in PSO for the DVRP [120].

For PSO, a multi-swarm algorithm based on the island model was proposed for the DVRP [130]. Particles from different swarms migrate regularly to others to maintain diversity. A different multi-swarm was introduced in [134], where the communication was performed only with the arrival of a new environment.

4.1.5. Hybridizations

The strategies in this category emphasize more on the hybridization of the SI algorithm with a local search method. Usually, algorithms enhanced by a local search, known as memetic algorithms, provide significantly better solutions with the price of increasing the computation time. The idea is to let the main algorithm provide an initial solution for the local search to optimize. But since local search operators generate strong exploitation the algorithms need to maintain diversity.

The P-ACO described above was improved in [44, 126], where several local search steps based on simple and adaptive inversions are applied to the iteration best ant. In addition the diversity is enhanced with the *triggered* random immigrants entering the memory, whenever all ants stored in the memory are identical. Recently, another memetic-based ACO was proposed in [35] that uses a local search whenever a new best solution is found. Whenever the algorithm is entering stagnation behaviour (very often because of the local search), the pheromone trails are re-initialized to the maximum value to maintain diversity.

4.2. SI in dynamic continuous optimization

The MPB described in Section 2.5 is the most widely used problem for DOPs in the continuous space. The key idea to tackle this problem (and in general multimodal problems as GDBP) is to locate and track all peaks (local optima) in the search space. When a change occurs, one of these local optima may become the global optimum. Other time varying functions include the Sphere, Ackley, Griewanks, Rastrigin, Rosenbrock, Schwefel, where a random value chosen from an exponential distribution is added/subtracted in every dimension. The Sphere and the two dimensional Rosenbrock functions are unimodal whereas the remaining functions are multimodal, just like the MPB and GDBP. Other time varying unimodal functions include the parabolic function.

From Table 3 it can be observed that PSO is mostly used on continuous problems. Each particle in the swarm has its own knowledge about its local environment. If the knowledge helps to solve the problem when a dynamic change occurs, it will quickly transfer among particles via simple interactions between particles. In this way, the position of the particles will be updated (essentially) towards the global optimum. Similarly, other SI algorithms can help to solve the problem but using different interactions. Nevertheless, all SI algorithms may need some enhancements in some perspective to escape from the previously converged optimum and effectively track the new one.

Several strategies were proposed to address this issue, which are categorized in the following subsections.

4.2.1. Increasing diversity after a change

Similarly, with the corresponding category of discrete optimization above, the strategies in this category depend on the detection of change.

In [198, 199] a conventional PSO algorithm was applied on the time-varying parabolic function that reflects a simple unimodal fitness landscape. When a dynamic change occurs, part of the population is randomly initialized.

4.2.2. Maintaining diversity during execution

Strategies in this this category explicitly maintain the diversity during the execution using different techniques, which are further grouped as follows.

1. Special mechanisms. Inspired from the atom field, Blackwell et al. [97] introduced a multi-charged PSO (mCPSO) and a multi-quantum swarm optimization (mQSO). In mCPSO, a part of particles in each swarm, named "charged" particles, repel each other and circle around neutral particles. In mQSO, the "quantum" particles move randomly around the best particle. Both algorithms were improved by re-initializing the worst performing swarm whenever all swarms converge for maintaining diversity [164]. Li et al. [150] integrated the quantum particles (from mQSO) to the SPSO to improve its adaptation capabilities. Thereafter, both SPSO and mQSO were further improved by converting particles to quantum particles for a single iteration after a dynamic change is detected [146]. The quantum principle introduced in [164] was also applied in a speciation FA (SFA) algorithm [179] and in a continuous variation of ACO [190].

   Another example is a BFO for dynamic environments [174], where chemotaxis is performed every iteration, reproduction is performed every f_r iterations, and elimination-dispersion is performed whenever a dynamic change occurs. In this way, bacteria will spread out to the search space, e.g., carrying out exploration.
2. Individual selection. Daneshyari and Yen [145] modified both the particle selection and replacement mechanisms so that the most diversified particles (measured by the Hamming distance) are selected and the particles that have close locations are replaced. Tang et al. [173] proposed a dynamic BFO (DBFO) that uses a more flexible selection scheme to maintain diversity, where the selection during reproduction is made probabilistically, in a way similarly to the one used in EAs. In [195], a modified ABC (MABC) was proposed with the restriction of the communication among bees to only the ones that are closer.
3. Operator selection. A PSO-based memetic algorithm [144] was proposed, where two local search operators are adaptively selected for maintaining the diversity.
4. Randomization. Kojima et al. [196] applied a dynamic ABC (DABC) algorithm to the dynamic Rastring and Rosenbrock functions, where independent food sources are used for each search phase and the cauchy distribution is taken in the employed phase.

In contrast, other strategies in this category focus on adjusting the neighborhood of an individual to implicitly maintain the diversity during the optimization process. For example, the speciation based PSO (SPSO), proposed by Parrott and Li [102, 165], adaptively organizes a neighbourhood of particles in each iteration according to certain principles based on their fitness and distance. In SPSO, the number and size of swarms are adaptively adjusted by constructing an ordered list of particles, ranked according to their fitness, with spatially close particles joining a particular species. The speciation scheme was adopted in a speciation FA (SFA) algorithm [179] to create sub-swarms.

Similarly, the compound concept derived by a branch of physics was introduced in PSO [166, 101]. Instead of the "fittest first" principle in SPSO, swarms are constructed based on a "worst first" principle, but each composite particle consists of three fixed particles. Thereafter, an opposite version was proposed, where a sub-swarm is generated based on the "fittest-oriented" principle [167].

Janson and Middendorf [162] proposed a partitioned hierarchical PSO (PH-PSO) that uses a dynamic tree-based neighbourhood structure. Each particle is placed on a single node of the tree and particles are arranged hierarchically according to their fitness. Chen et al. [169] have used a bee life cycle method to dynamically adjust the size of the colony according to the local fitness landscape during optimization. Parsopoulos and Vrahatis [193] proposed a unified PSO with a constriction factor, which combines the global and local best PSO variants.

Furthermore, ACO variations that replace the conventional discrete probability distribution by a continuous are applied on different time-varying functions [191] and GDBG [184, 185]. By setting a higher evaporation rate ACO variations can adapt to changes as described in Section 4.1.

4.2.3. Memory schemes

As mentioned above, the memory scheme aims to transfer "useful" knowledge from previous search to accelerate the current search. This scheme can also help maintain the diversity.

To implement a memory scheme, an external archive is normally needed. An archive population was used to store good solutions whenever a dynamic change is detected or when a peak is found in a tri-island model [141]. Best solutions are maintained in an archive whenever a change occurs in the employed bee phase for a differential ABC algorithm [197] and an improved ABC algorithm [171]. A history-driven SFA (HdSFA) [180] algorithm uses two types of memory: a short-term memory, which is responsible for storing information regarding the current environment, and a long-term memory, which is responsible to store information regarding the previous environments. The same memory scheme is adopted in PSO [181]. Recently, SPSO was enhanced with a memory that stores particles from different species [168].

4.2.4. Multiple population methods

The multi-swarm scheme is a kind of divide-and-conquer strategy. Searching in a different area for each swarm means that the search space is divided into multiple sub-spaces and

each swarm is responsible for searching one sub-space. By doing so, it has several advantages [200]. Firstly, the global diversity can be maintained since swarms search in different areas. Secondly, tracking multiple optima simultaneously is possible. Thirdly, any scheme based on a single swarm approach can be easily extended to a multi-swarm version. Multi-swarm strategies can be classified to the following categories in terms of the number of swarms.

1. Static (or fixed) number of swarms. This group of strategies can be further classified to two types: competing and collaborative types. For the competing type, swarms compete with each other, i.e., they are not allowed to search in the same area. On the contrary, in the collaborative type, swarms are allowed to search in the same area and cooperate with each other, i.e., they may share information. Blackwell's mCPSO and mQSO [164] belong to the competing type. Each swarm has an exclusion radius to prevent sub-swarms from overlapping each other. Similar ideas of the exclusion principle in mQSO were adopted in other algorithms, such as the multi-swarm modified AFSO (MAFSO) [176, 177], multi-swarm FA algorithms [178, 182, 183], multi-BFO (MBFO) algorithm [175], multi-swarm ABC [188], and k-means clustering ABC (CABC) [189]. A typical collaborative type of multi-swarm algorithm is the collaborative evolutionary swarm optimization (CESO) [157], where two heterogeneous swarms cooperate with each other. The two swarms follow the crowding differential evolution (CDE) [201] and PSO models, respectively. The former swarm is responsible for diversity maintenance whereas the latter swarm is responsible for tracking the global optimum. Similarly, the dual-swarm approach was adopted in [192, 155]. An evolutionary swarm cooperative algorithm (ESCA) with an additional swarm, was proposed in [154]. Within ESCA, three swarms are used and share information among each other.
2. Dynamic number of swarms. This type of strategies can be further grouped to two models in terms of the way to create sub-swarms: regrouping and splitting models. The regrouping model regroups individuals by a certain means every iteration or when a certain criterion is satisfied, such as the speciation based PSO (SPSO) [102, 165] and the randomized regrouping multi-swarm PSO [194]. The clustering PSO (CPSO) algorithm proposed by Li and Yang [149, 99] uses a hierarchical clustering method to generate sub-swarms whenever a change is detected. To avoid change detection, CPSO was later on improved with a new version, called CPSOR [98], where random particles are introduced and clustered into new swarms when the total number of particles drops to a threshold ratio. The splitting model normally generates sub-swarms by splitting off from a main swarm when a certain criterion is met, such as the fast multi-swarm optimization (FMSO) algorithm [159]. FMSO starts with a parent swarm. Whenever the best particle gets better, a child swarm is generated with the best particle and particles within a given distance from the best one. Similar ideas were adopted in a multi-swarm PSO algorithm (mPSO) [158], and algorithms proposed in [152, 151].
3. Adaptive number of swarms. To efficiently solve DOPs by the multi-swarm scheme, one key issue is to adapt the number of swarms [200, 163, 164]. This issue becomes more challenging for DOPs with a changing number of optima. The basic idea is to increase swarms when current swarms are converging and remove swarms when they are over-crowded. Blackwell [163] proposed the self-adaptive multi-swarm optimizer (SAMO) which is the first adaptive method regarding the number of populations. It begins with a single free swarm (e.g., a non-converging population). Free swarms gradually transform themselves into converging swarms. Converging swarms are identified when their radius is less than a given convergence radius r_{conv}). A new free swarm is randomly generated if there is no free swarm. This way, the number of populations is able to adaptively change based on the status of current swarms. An adaptive multi-swarm optimizer (AMSO) was proposed [148], where the number of swarms is adjusted according to the differences of the number of swarms between two successive "checking points". A multi-swarm AFSO was proposed [172], where a child swarm is generated whenever a parent swarm locates a new peak. The child swarm tracks the peak has been located. Recently, the number of swarms was adapted [147] using a database that collects heuristic information of the algorithm behaviour changes to better track multiple optima. Nseef et al. [170] proposed a multi-colony ABC algorithm. The number of colonies is adaptively maintained based on the dynamic change strength.

4.2.5. Hybridizations

The hybridization with different domain knowledge, e.g., genetic algorithms, differential evolution, or other meta-heuristic methods, e.g., microbial life, fuzzy, and cellular automata, is also an important research topic.

1. Hybridization with other domain knowledge. For example, a multi-strategy ensemble PSO (MEPSO) [161] uses Gaussian local search and differential mutation for exploration and exploitation. A dynamic macro-mutation operator was introduced in a PSO algorithm [160]. A PSO-based memetic algorithm [142] uses a ring topology structure and a fuzzy cognition. A fuzzy cognition with multiple local searches was also used in [143]. An improved mQSO algorithm (mQSOE) [186] uses an evaporation mechanism as used in ACO to penalize the fitness value of the best position found by each particle in the past.
2. Hybridization with other meta-heuristic methods. Hashemi and Meybodi [140, 156] proposed a hybrid model of PSO with cellular automata to address DOPs, where the search space is embedded with cellular automata and is separated into several cells. Each cell is allowed to contain only a specified number of particles to prevent diversity loss. Rezazadeh et al. [153] proposed

an adaptive PSO algorithm. The algorithm uses a fuzzy C-means mechanism to adapt the exclusion radii and inertia weight parameters. Karimi et al. proposed particles inspired from the microbial life that can reproduce new infant particles to replace the old particles.

4.3. SI in dynamic constrained optimization

Dynamic constrained optimization problems are challenging because they often affect the feasibility of the algorithmic solutions. Liu [77] transformed the problem into a series of static constrained optimization problems of different periods. Each designed static problem is assigned with an individual fitness function based on the original objective and its constraints. The PSO algorithm is then applied to each designed problem without the need to take care of the feasibility of the particles. The experimental results showed that this PSO algorithm can find the optimal solutions on a set of benchmark problems.

Differently, Dewan and Nayak [202] used a penalty based function with PSO to handle infeasible particles whereas new types of particles for local search are introduced for feasible particles. The algorithm was tested on a known benchmark set and was compared with the results of other EAs. The PSO results are quite competitive in comparison with EAs.

Yin and Sun [203] proposed a novel adaptive mutation PSO (AMPSO) algorithm based on a generalized dynamic constraints satisfaction strategy. The mutation operator within AMPSO is applied to the inactive (e.g., particles that have made no progress for a number of iterations) or less fit particles of the swarm according to an adaptively changing mutation probability.

Bu et al. [56] applied a variation of the SPSO with an ensemble of strategies to locate and track feasible regions. The strategies include: a gradient-based repair, an adaptive local search, constraint handling, memory and prediction strategy.

4.4. SI in dynamic multi-objective optimization

Dynamic multi-objective optimization problems are more challenging than DOPs (single-objective) because a set of Pareto optimal solutions needs to be tracked rather than a single optimal solution. Wei and Wang [204] proposed a hyper rectangle search based PSO algorithm to tackle such problems. The algorithm utilizes a hyper rectangle search to approximately predict the next optimal solutions. The experimental results on a known benchmark set showed that PSO can effectively locate and track the set of Pareto optimal solutions over time.

Differently, Wei and Jia [205] integrated a new points selection strategy with PSO to generate the initial swarm of the next time interval, which acts as a restart strategy. In addition, an attraction based local search operator is incorporated to utilize the information of the swarm.

A multi-swarm PSO was proposed where each swarm solves a single objective function independently and communicates with other swarms to transfer the knowledge [206, 207]. In order to maintain diversity, a percentage of the swarm are reinitialized whenever a dynamic change is detected by sentry particles.

4.5. SI in dynamic classification

Several PSO algorithms, e.g., QPSO, CPSO and CCPSO, proposed in dynamic continuous optimization, were applied for training supervised feed-forward neural networks in classification problems [208, 209]. The dynamic changes during training occur to the decision boundaries, known as concept drift. Since training a neural network is a continuous optimization problem the aforementioned algorithms were applied directly. The experimental results showed that QPSO, CPSO and CCPSO had better classification error than a PSO with a restart and a gradient descent algorithm, e.g., back propagation algorithm.

4.6. Discussions

Every strategy for adapting SI to dynamic environments has advantages and limitations. For example, the diversity maintaining/increasing strategy can well address the diversity loss issue, but it is hard to know the optimal frequency for diversity increasing and to design efficient diversity maintaining mechanisms. Memory schemes are able to help an algorithm to quickly adapt to new environments, but it does not help in environments with significant changes. The multi-swarm strategy is able to maintain the diversity at the global level, but the optimal number of swarms and the search area of each swarm are very hard to configure.

Due to the advantages and limitations of different strategies aforementioned, multiple strategies are usually considered in most studies. For example, in many multi-swarm algorithms, diversity maintaining and memory strategies are commonly used to enhance the performance. No matter what strategies are used, the balance of knowledge transfer and diversity maintenance should always be considered for handling environmental changes.

5. Real-world applications of SI in dynamic environments

In the this section real-world applications that appear in SIDO are reviewed and classified into discrete and continuous applications. The difference between "real-world applications" and the applications discussed in Tables 2 and 3 is rather arbitrary [210]. In this paper, the applications discussed in Tables 2 and 3 focus on well-known models that aim to model a real-world problem or setting (e.g., multi-objective or constraint handling). Such models may provide an indication whether such an application may work in the real-world. On the other hand, the real-world applications discussed in the following sections focus in applications that were used in some industries or were tested on real data.

5.1. Discrete applications

The field of network routing has been extensively studied by ACO algorithms. Generally, the routing problem in networks is defined as the problem of defining path flows to forward incoming data traffic such that the overall network performance is maximized. According to the characteristics of the processing and transmission components, the traffic flow pattern, and the performance expected to be delivered of the

network, different types of routing problems can be defined. Mainly, there are two categories of network routing problems in which ACO has been applied: *wired*, e.g., wired best effort networks [211, 212, 213, 214, 215] and wired quality of service (QoS) networks [216, 217, 218, 214, 219, 220], and *wireless*, e.g., mobile ad hoc networks [221, 222, 223, 224, 225]. The telecommunication algorithms simply rely on the adaptation capabilities of ACO.

Liu et al. [226] proposed a multicast routing protocol based on AFSO to address the latency and bandwidth constraints of QoS networks. The simulation results showed that AFSO achieves promising performance and robustness in real-time multimedia transmission networks. ABC algorithms were also studied for mobile ad hoc networks [227, 228] and PSO for peer-to-peer networks [229, 230].

Vogel et al. [231] proposed an ACO algorithm to address dynamic changes in the production system. A Position-Operation-Pheromone matrix is maintained to store the operations. Whenever a new job arrives, the pheromone values are re-initialized. Their experiments on a real-world data collection showed that the proposed ACO method outperforms the manual method and a traditional priority-rule method. In contrast, the ACO method was outperformed by an EA.

Silva and Runkler [68] addressed the cash machine problem using a conventional ACO algorithm. Basically, the problem is that a cash logistics company needs to fill the cash containers according to their money levels for a number of cash machines of a specific bank in Frankfurt. The results showed that ACO is able to automatically output solutions with the minimum travelling time, considering the distances between the cash machines and their money levels. However, there are no comparisons with any other automatic or manual methods.

Nakrani and Tovey [232] used a new bee algorithm (e.g., honey bee algorithm development) for the dynamic server allocation in an internet server colony. Foraging bees and flower patches in their model represent the servers and HTTP dynamic request queues, respectively, in an internet server colony. Their experimental results on cases of a hosting center with two and three virtual servers composed from a total of fifty servers showed that the honey bee algorithm performs better than other traditional algorithms, especially in highly dynamic and unpredictable internet environments, due to its adaptation capabilities.

Triwate and Luangpaiboon [233] applied a bee algorithm (e.g., bee algorithm development) to minimize the imbalance of a dynamic multi-zone dispatching system, where each bee represents a dispatching pattern. The simulation study was based on real-world data from Thai local transportation firms. The authors concluded that the bee algorithm reaches the optimum very fast on small problems but requires more time for medium and large size problems.

Wedde et al. [234] used a bee algorithm (e.g., BeeHive algorithm development) to route vehicles on the roads with lower traffic jams and congestions to minimize transportation times. The topological road data of eastern German Ruhr District was considered and several traffic scenarios were generated. The comparison with a traditional shortest-path algorithm (e.g., Dijkstra algorithm) showed that the BeeHive algorithm has better congestion handling. This is because the time-linkage of the problem is considered by BeeHive, which routes vehicles on different paths dynamically, whereas Dijkstra's algorithm always selects the shortest one. The authors showed that after some time heavy congestions occur with Dijkstra-based routing, whereas Bee-based routing remains congestion free.

Sulaiman et al. [235] used an FA to solve the optimal allocation and sizing problem of distributed generation units in distribution networks. For the experiments an IEEE 69-bus distribution test system was used. Their results show that the determination of the optimal location and size of distributed generation units reduces power losses and improves the voltage profile of the power system. In addition, the comparison of the proposed FA with an EA showed comparable results.

Teodorović and Dell'Orco [236] applied a bee algorithm (e.g., bee colony optimization development) with fuzzy logic to the ride-matching problem. The problem is to optimize the routing and scheduling of vehicles and passengers for a given period of time. The uncertainties in this application appear by possible delays caused by the passengers or by traffic congestion. The authors considered the ride sharing demand of 97 travellers from Trani, a small city in Italy. The results obtained by the algorithm are very promising.

Garcia et al. [237] uses an ACO to solve the problem of path planning for mobile robots with obstacle avoidance. The workspace (search space) is discretized in a matrix of 50×50 nodes, where the mobile robot is able to navigate and build paths. PSO is used for the same problem with moving obstacles [238].

Li et al. [239] applied a discrete version of PSO to the resource allocation problem in a cooperative OFDMA system to maximize the average utility of all mobile stations under multi-service. The correlation between adjacent frames is exploited to transfer knowledge. Their results proved that knowledge transfer reduced the computation complexity significantly and achieved better performance in terms of maximizing utility.

Hsu [240] hybridized a PSO with an event-based heuristic to solve two essential seaside operations planning problems, i.e., berth allocation and quay crane assignment problems, simultaneously. The key idea of the approach is to support variable-in-time quay crane assignment rather than time-invariant. In this way, the assignment of quay cranes is utilized because they can be adjusted accordingly.

Eaton et al. [241] uses an ACO to reschedule trains at junctions and stations when train delays occur. A new feasible schedule is generated based on the previous infeasible one using a path-preserving heuristic.

5.2. Continuous applications

The dynamic economic dispatch (DED) problem has been extensively studied by SI algorithms such as BFO [242], PSO [243, 244, 245, 246], FA [247], AFSO [248] and ABC algorithms [249, 250, 251, 252, 253]. Moreover, a hybrid SFLA algorithm [254] outperformed PSO- and ABC-based algorithms. The main objective of the DED problem is to reduce the opera-

tional costs in an electrical power system without violating a set of various security and efficiency constraints simultaneously.

Niknam et al. [255] used a bee algorithm (e.g., honey bee mating optimization development) to solve the dynamic optimal power flow[4] and minimize the total system generation cost. The proposed bee algorithm differs from the original development of the bee algorithm because it uses a mutation operator to increase the population diversity. The experimental results on 14-,30-, and 118-bus test systems showed that this mutation improves the quality of the conventional bee algorithm.

Tang et al. [256, 257] proposed a BFO algorithm to solve the optimal power flow problem with power system dynamic loads. The reproduction process of the BFO uses a selection scheme based on the roulette wheel method to maintain diversity. The dispersion and elimination processes are not considered in order to avoid too much randomization. An IEEE 30- and 118-bus test system is used for the experiments. The proposed BFO algorithm was compared with conventional BFO and PSO on three different levels of load changes, showing satisfactory performance.

Takano et al. [258] used an ABC algorithm to coordinate rescue agents in dynamic disaster environments, e.g., a city after an earthquake occurrence. The algorithm is modified to only communicate with nearby bees. The experiments on the RoboCup Rescue Simulation System showed that ABC can rescue all victims quickly.

FA [259] and PSO [260] were used to tune the proportional integral derivative controller for an automatic voltage regulator system. The optimal tuning of the controller is based on the time-variability of real-world power system operation. Comparisons between the recent FA and PSO algorithms showed that the former is more robust than the latter. BFO [261] was also used for the tuning of the controller and was more robust and efficient than an EA.

Gomes [262, 263] used PSO and FA algorithms as optimization engines on structural mass optimization. The structural optimization of shape and size is considered as highly non-linear. The problem contains dynamic constraints that may affect the fitness function. Hence, a penalty function technique is used to address violations. Three examples of 10-, 37-, and 128-truss bar structures were taken into account and showed that FA performed slightly worse than PSO but both SI algorithms performed better than traditional optimization methods.

ACO and ABC were applied to the dynamic load balancing problem [264]. The main objective of the problem is to find the optimal allocation of workloads among systems, e.g., reduce the load from busy resources and transfer them to idle resources. The experimental study on a cluster of four machines and the Amazon EC2 cloud showed that the ABC is faster than ACO and other traditional algorithms.

A BFO algorithm was applied to determine the shortest feasible path from a current position to the target position in a 2D space with moving obstacles [265]. The algorithm is able to avoid obstacles and find a path towards the target position. Similarly, PSO algorithms showed that they are also able to plan an optimal path for a robot by avoiding moving obstacles [266, 267, 268].

A contaminant source identification problem in water distribution networks is a nonlinear programming problem. The aim of the problem is to search for the location and the time history of the contaminant according to the observed data up to the current time. Liu et al. [269] used a multi-swarm based PSO algorithm to solve this problem.

6. Conclusions and future work

This paper attempts to review the related work of SIDO found from several web-search engines. The most important applications of SIDO are classified into continuous or discrete problems. The strategies used to enhance SI algorithms to cope with dynamic changes are grouped and extensively discussed. In addition, SIDO real-world problems are reviewed in this paper.

The review of this paper was constructed as follows. The search was conducted from five recognized scientific databases, i.e., IEEExplore[5], Science Direct[6], SpringerLink[7], Scopus[8] and Google Scholar[9] using terms like "ant colony optimization" AND ("dynamic environment" OR "dynamic optimization" OR "dynamic function" OR "time-varying" OR "dynamic problem" OR "dynamic routing" OR "dynamic scheduling" OR "dynamic multi-objective" OR "dynamic constraint") for ACO. The same search format was used for PSO with "particle swarm optimization" instead and similarly for the other SI algorithms. The bibliography retrieved includes journal articles, conferences papers, book chapters and technical reports. Then, the results were categorized by: (a) SI algorithms in Fig. 3 and (b) year of publication in Fig. 4.

From Fig. 3, it can be observed that ACO and PSO have attracted more attention than the other SI algorithms in SIDO. This is natural because they were proposed much earlier than other SI algorithms. It may also be the case that they are more effective than the recently proposed SI algorithms. This is because in many cases the SI algorithms were using core components from these algorithms. For example, the local best position component of the particle introduced in PSO was used as an enhancement strategy in many cases in ABC, AFSO, FA and BFO. From Fig 4, it can be observed that roughly the field of SIDO began to grow in 2001 and gradually grew until 2009. Since then the field has its ups and downs regarding the number of publications.

In general, all SI algorithms have been mainly applied on dynamic versions of the corresponding stationary optimization problems that they were initially developed. For example, ACO is mainly used in dynamic discrete problems that are modelled

[4] The optimal power flow problem may also contain discrete variables.

[5] http://ieeexplore.ieee.org
[6] http://www.sciencedirect.com
[7] http://link.springer.com
[8] http://www.scopus.com
[9] http://scholar.google.com

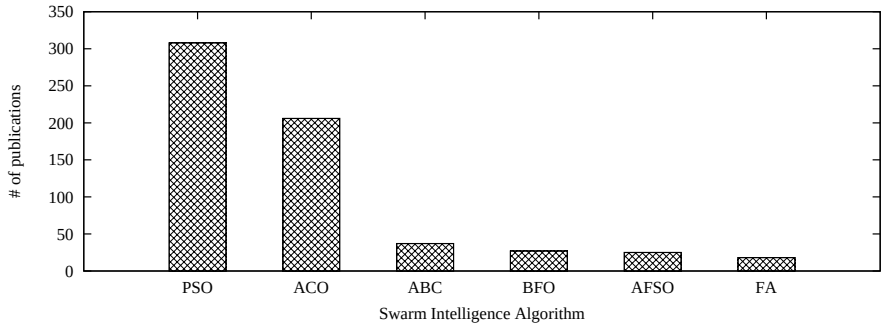

Fig. 3: Distribution of publications sorted by swarm intelligence algorithms.

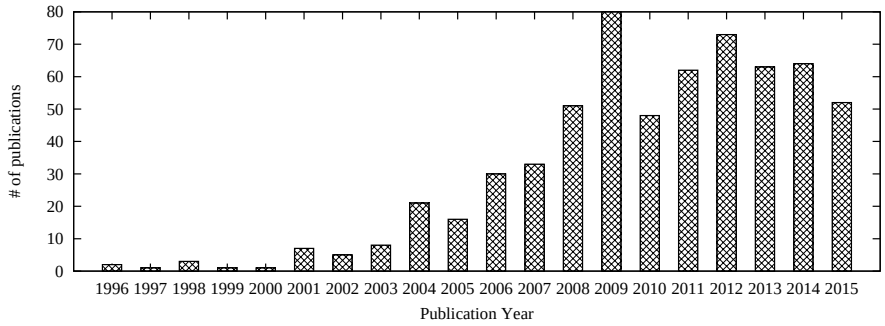

Fig. 4: Distribution of publications sorted by year of publication.

by graphs because it is a graph-search algorithm. Also, the enhancement strategies integrated to the different SI algorithms usually improve their performance when the changing environments are similar. Finally, a local search operator is a core component of SI algorithms when addressing DOPs. SI algorithms perform global optimization and they are not accurate in their output. Hence, a local search operator can significantly improve the quality of the output. However, the integration of local search in SI algorithms has to be done in such a way that does not significantly increase the computation time or waste evaluations [35].

Although the research in SIDO showed significant increase, it still has several aspects for future directions that may contribute to further grow the field of SIDO, which are summarized as follows:

- Experimentation protocols: Since there is not any agreed computational experimentation protocol yet, it may be necessary to define different experimentation protocols for different problem classes, e.g., continuous, discrete and constrained problems, due to the different characteristics regarding both the methodologies used and the search space.

- Benchmark generators: The fields of dynamic multi-objective and dynamic constrained optimization are still young. Most benchmark generators used are simply extensions from the static instances. It may be necessary to develop benchmark generators especially for these recent fields to compare new algorithms developed.

- Avoiding change detection: Change detection methods by re-evaluating individuals cannot always guarantee a successful detection, especially in the case where a change does not affect current solutions in the fitness landscape. Such situations may happen in dynamic environments where a part of the fitness landscape changes. In addition, such detection methods will never work in dynamic environments with noise since noise will be misinterpreted as changes in every re-evaluating operation. Therefore, avoiding change detection is needed to handle changes more effectively.

- Modelling real-world scenarios: It will be interesting to consider other research areas in dynamic optimization that have attracted less interest and model many real-world scenarios, e.g., constrained DOPs, dynamic multi-objective problems or even dynamic constrained multi-objective problems.

- Consideration of other prospectives: The tracking moving optimum (TMO) framework is usually used in SIDO to tackle DOPs. Recently, a new perspective on DOPs has been established, known as robust optimization over time (ROOT), where the target is to find the chain of solutions which are robust over time [46]. Particularly, a solution is robust over time when its quality is acceptable to the changing environments during a predefined time interval. In fact, problems in scheduling and vehicle routing have the time-linkage property. Therefore, the TMO framework may not be the best choice, because a decision that improves the performance at present may affect the perfor-

mance in the future. As a result, the overall performance may be degraded in the long run. So far, ROOT was integrated only with PSO.

- Choice of algorithms: So far, many SI algorithms have been proposed in the last 30 years. For a specific problem, most readers are not sure what algorithms to choose. This is because making a right choice is again an optimization problem. Experience and trail-and-error based choice are still the major ways to make the choice. Therefore, finding the right algorithms most suitable for a given problem or problems with a certain type of properties would also be an interesting topic in SIDO in the future.

- Theoretical development: Usually, the performance of the algorithms in DOPs is analyzed empirically, rather than theoretically, due to the mathematical challenges of SI algorithms, or metaheuristics in general. Although theoretical works exist the research area is still limited. Theoretical development in SIDO should be enhanced significantly in order to have a deep understanding on why and how SI algorithms work in DOPs.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1, by the National Natural Science Foundation of China (NSFC) under Grants 61673331 and 61673355, by the Hubei Provincial Natural Science Foundation of China under Grant 2015CFA010, and by the 111 project under Grant B17040.

References

[1] G. Beni, The concept of cellular robotic system, in: 1988 Proceedings of the IEEE International Symposium on Intelligent Control, 1988, pp. 57–62.

[2] G. Beni, J. Wang, Swarm intelligence, in: Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan, RSJ Press. Tokyo, 1989, pp. 425–428.

[3] G. Beni, S. Hackwood, Stationary waves in cyclic swarms, in: Proceedings of the 1992 IEEE International Symposium on Intelligent Control, 1992, pp. 234–242.

[4] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the 1999 IEEE Congress on Evolutionary Computation, vol. 3, 1999, pp. 1875–1882.

[5] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in: F. Vaerla, P. Bourgine (Eds.), Proceedings of the European Conference on Artificial Life, Elsevier Publishing, Paris, 1991, pp. 134–142.

[6] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948

[7] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Technical Report TR06, Erciyes University, Turkey, 2005.

[8] K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst. Mag. (2002) 52–67.

[9] X.-S. Yang, Firefly algorithm, Nature-Inspired Metaheuristic Algorithms 20 (2008) 79–90.

[10] X.-S. Yang, Firefly algorithms for multimodal optimization, in: O. Watanabe, T. Zeugmann (Eds.), Stochastic Algorithms: Foundations and Applications, Vol. 5792 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 169–178.

[11] L. Li, Z. Shao, J. Qian, An optimizing method based on autonomous animals: fish-swarm algorithm, Sys. Eng. Theory Pract. 22(11) (2002) 32–38.

[12] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments–a survey, IEEE Trans. Evol. Comput. 9 (3) (2005) 303–317.

[13] C. Cruz, J. R. Gonzalez, D. A. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, Soft Comput. 15 (7) (2011) 1427–1448.

[14] T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evol. Comput. 6 (2012) 1–24.

[15] S. Yang, Y. Jiang, T. Nguyen, Metaheuristics for dynamic combinatorial optimization problems, IMA J. Manag. Math. 24 (4) (2013) 451–480.

[16] E. Alba, A. Nakib, P. Siarry (Eds.), Metaheuristics for Dynamic Optimization, Vol. 433 of Studies in Computational Intelligence, Springer, Heidelberg, 2013.

[17] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer, Dordrecht, 2001.

[18] S. Yang, Y.-S. Ong, Y. Jin (Eds.), Evolutionary computation in dynamic and uncertain environments, Studies in Computational Intelligence, vol. 51 Springer, Heidelberg, 2007.

[19] S. Yang, X. Yao (Eds.), Evolutionary Computation for Dynamic Optimization Problems, Vol. 490 of Studies in Computational Intelligence, Springer, Heidelberg, 2013.

[20] R. Morrison, Designing Evolutionary Algorithms for Dynamic Environments, Springer-Verlag, Berlin Heidelberg, 2004.

[21] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, Appl. Soft Comput. 13 (10) (2013) 4023–4037.

[22] P. Bosman, Learning, anticipation and time-deception in evolutionary online dynamic optimization, in: Proceedings of the 2005 Genetic and Evolutionary Computation Conference, ACM, New York, NY, 2005, pp. 39–47.

[23] T. Nguyen, X. Yao, Dynamic time-linkage problems revisited, in: M. Giacobini, A. Brabazon, S. Cagnoni, G. Di Caro, A. Ekárt, A. Esparcia-Alcázar, M. Farooq, A. Fink, P. Machado (Eds.), Applications of Evolutionary Computing, Vol. 5484 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 735–744.

[24] M. Helbig, A. Engelbrecht, Dynamic multi-objective optimization using PSO, in: E. Alba, A. Nakib, P. Siarry (Eds.), Metaheuristics for Dynamic Optimization, Vol. 433 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2013, pp. 147–188.

[25] M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: test cases, approximations, and applications, IEEE Trans. Evol. Comput. 8 (5) (2004) 425–442.

[26] T. Nguyen, X. Yao, Continuous dynamic constrained optimization-the challenges, IEEE Trans. Evol. Comput. 16 (6) (2012) 769–786.

[27] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.

[28] J.E. Rowe, M.D. Vose, A.H. Wright, Reinterpreting No Free Lunch, Evol. Comput. 17(1) (2009) 117-129.

[29] A. Auger and O. Teytaud, Continuous Lunches are Free!, Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO-2007, 2007, pp 916-922.

[30] A. Auger, O. Teytaud, Continuous lunches are free plus the design of optimal optimization algorithms, Algorithmica 57 (1) (2010) 121–146.

[31] A. Alabert, A. Berti, R. Caballero, M. Ferrante, No-free-lunch theorems in the continuum. Theoret. Comput. Sci. 600 (2015) 98-106.

[32] W. Rand, R. Riolo, Measurements for understanding the behavior of the genetic algorithm in dynamic environments: a case study using the shaky ladder hyperplane-defined functions, in: Proceedings of the 2005 Genetic and Evolutionary Computation Conference, ACM, New York, NY, 2005, pp. 32–38.

[33] M. Mavrovouniotis, S. Yang, Ant algorithms with immigrants schemes for the dynamic vehicle routing problem, Inf. Sci. 294 (2015) 456–477.

[34] R. Morrison, Performance measurement in dynamic environments, in: Proceedings of the 2003 Genetic and Evolutionary Computation Conference, 2003, pp. 5–8.

[35] M. Mavrovouniotis, F. M. Müller, S. Yang, Ant colony optimization with local search for the dynamic travelling salesman problems, IEEE Trans. Cybern. PP (99) (2016) 1–14. doi:10.1109/TCYB.2016.2556742.

[36] M. Mavrovouniotis, S. Yang, X. Yao, Multi-colony ant algorithms for the dynamic travelling salesman problem, in: 2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014, pp. 9–16.
[37] M. Mavrovouniotis, S. Yang, Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem, in: 2012 IEEE Congress on Evolutionary Computation (CEC), 2012, pp. 2645–2652.
[38] A. Younes, O. Basir, P. Calamai, Benchmark generator for dynamic optimization, WSEAS Trans. Syst. 3 (1), 2004, 273–278.
[39] M. Mavrovouniotis, F. M. Müller, S. Yang, An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem, in: Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO15), ACM Press, New York, NY, 2015, pp. 49–56.
[40] C. Li, S. Yang, T. Nguyen, E. Yu, X. Yao, Y. Jin, H.-G. Beyer, P. Suganthan, Benchmark generator for CEC'2009 competition on dynamic optimization, Technical Report, Department of Computer Science, University of Leicester, UK, 2009.
[41] K. Weicker, Performance measures for dynamic environments, in: J. Merelo-Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, J.-L. Fernández-Villacañas (Eds.), Parallel Problem Solving from Nature – PPSN VII, Vol. 2439 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 64–73.
[42] R. Salomon, P. Eggenberger, Adaptation on the evolutionary time scale: A working hypothesis and basic experiments, in: J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), Artificial Evolution, Vol. 1363 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 251–262.
[43] K. Weicker, N. Weicker, On evolution strategy optimization in dynamic environments, in: Proceedings of the 1999 IEEE Congress on Evolutionary Computation CEC 99, vol. 3, 1999, pp. 2039–2046.
[44] M. Mavrovouniotis, S. Yang, A memetic ant colony optimization algorithm for the dynamic travelling salesman problem, Soft Comput. 15 (7) (2011) 1405–1425.
[45] X. Yu, Y. Jin, K. Tang, X. Yao, Robust optimization over time - a new perspective on dynamic optimization problems, in: 2010 IEEE Congress on Evolutionary Computation (CEC), 2010, pp. 3998–4003.
[46] Y. Jin, K. Tang, X. Yu, B. Sendhoff, X. Yao, A framework for finding robust optimal solutions over time, Memetic Comput. 5 (1) (2013) 3–18.
[47] B. Moghaddam, R. Ruiz, S. Sadjadi, Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm, Comput Ind. Eng. 62 (1) (2012) 306–317.
[48] N. Toklu, R. Montemanni, L. Gambardella, An ant colony system for the capacitated vehicle routing problem with uncertain travel costs, in: 2013 IEEE Symposium on Swarm Intelligence (SIS), 2013, pp. 32–39.
[49] N. Toklu, R. Montemanni, L. Gambardella, A multiple ant colony system for a vehicle routing problem with time windows and uncertain travel times, J. Traffic Logist. Eng. 2 (1) (2014) 52–58.
[50] M. Mavrovouniotis, S. Yang, An immigrants scheme based on environmental information for ant colony optimization for the dynamic travelling salesman problem, in: J.-K. Hao, P. Legrand, P.and Collet, N. Monmarché, E. Lutton, M. Schoenauer (Eds.), Artificial Evolution, Vol. 7401 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 1–12.
[51] N. Mori, S. Imanishi, H. Kita, Y. Nishikawa, Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm, in: T. Bäck (Ed.), International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1997, pp. 299–306.
[52] R. Morrison, K. D. Jong, Measurement of population diversity, in: P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, M. Schoenauer (Eds.), Artificial Evolution, Vol. 2310 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 31–41.
[53] E. Jen, Stable or robust? whats the difference?, Complexity 8 (3) (2003) 12–18.
[54] M. Guntsch, M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic TSP, in: E. J. W. Boers (Ed.), Applications of Evolutionary Computing, Vol. 2037 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 213–222.
[55] M. Mavrovouniotis, S. Yang, Adapting the pheromone evaporation rate in dynamic routing problems, in: A. Esparcia-Alcázar (Ed.), Applications of Evolutionary Computation, Vol. 7835 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 606–615.
[56] C. Bu, W. Luo, L. Yue, Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies, IEEE Trans. Evol. Comput. PP (99) (2016) 1–1. doi:10.1109/TEVC.2016.2567644
[57] C.-K. Goh, K. Chen T., An investigation on noisy environments in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 11 (3) (2007) 354–381.
[58] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evol. Comput. 8 (2) (2000) 173–195.
[59] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.
[60] M. Câmara, J. Ortega, F. Toro, Parallel processing for multi-objective optimization in dynamic environments, in: 2007 IEEE International Parallel and Distributed Processing Symposium, 2007, pp. 1–8.
[61] C.-K. Goh, K. Chen T., A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization, IEEE Trans. Evol. Comput. 13 (1) (2009) 103–127.
[62] P. Rohlfshagen, P. Lehre, X. Yao, Dynamic evolutionary optimization: an analysis of frequency and magnitude of change, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO '09, ACM, New York, NY, 2009, pp. 1713–1720.
[63] P. Rohlfshagen, X. Yao, Dynamic combinatorial optimisation problems: An analysis of the subset sum problem, Soft Comput. 15 (9) (2011) 1723–1734.
[64] J. Lewis, E. Hart, G. Ritchie, A comparison of dominance mechanisms and simple mutation on non-stationary problems, in: A. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN V, Vol. 1498 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 139–148.
[65] C. Eyckelhof, M. Snoek, Ant systems for a dynamic TSP, in: M. Dorigo, G. Di Caro, M. Sampels (Eds.), Ant Algorithms, Vol. 2463 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 88–99.
[66] M. Guntsch, M. Middendorf, Applying population based ACO to dynamic optimization problems, in: M. Dorigo, G. Di Caro, M. Sampels (Eds.), Ant Algorithms, Vol. 2463 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 111–122.
[67] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, A. V. Donati, Ant colony system for a dynamic vehicle routing problem, Combinat. Optim. 10 (2005) 327–343.
[68] C. Silva, T. Runkler, Ant colony optimization for dynamic traveling salesman problems, in: ARCS Workshops, 2004, pp. 259–266.
[69] A. Younes, P. Calamai, O. Basir, Generalized benchmark generation for dynamic combinatorial problems, in: Proceedings of the 2005 Genetic and Evolutionary Computation Conference, ACM, New York, NY, 2005, pp. 25–31.
[70] S. Yang, Non-stationary problem optimization using the primal-dual genetic algorithm, in: Proceedings of the 2003 IEEE Congress on Evolutionary Computation CEC '03, vol. 3, 2003, pp. 2246–2253.
[71] M. Mavrovouniotis, S. Yang, X. Yao, A benchmark generator for dynamic permutation-encoded problems, in: C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, M. Pavone (Eds.), Parallel Problem Solving from Nature - PPSN XII, Vol. 7492 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 508–517.
[72] R. Morrison, K. D. Jong, A test problem generator for non-stationary environments, in: Proceedings of the 1999 IEEE Congress on Evolutionary Computation CEC 99, vol. 3, 1999, pp. 2047–2053.
[73] R. Zhou, A. Nee, H. P. Lee, Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems, Int. J. Prod. Res. 47 (11) (2009) 2903–2920.
[74] S. Lorpunmanee, M. Sap, A. Abdullah, C. Chompoo-Inwai, Ant colony optimization for dynamic job scheduling in grid environment, Int. J. Comput. Inf. Eng. 1 (8) (2007) 469–476.
[75] M. Randall, A dynamic optimisation approach for ant colony optimisation using the multidimensional knapsack problem: recent advances in artificial life, Adv. Nat.l Comput. 3 (2005) 215–226.
[76] M. Mavrovouniotis, S. Yang, Applying ant colony optimization to dynamic binary-encoded problems, in: A. Mora, G. Squillero (Eds.), EvoApplications 2015: Applications of Evolutionary Computation, Vol.

9028 of Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 845–856.
[77] C. Liu, New dynamic constrained optimization PSO algorithm, in: Fourth International Conference on Natural Computation. ICNC '08, vol. 7, 2008, pp. 650–653.
[78] T. Nguyen, X. Yao, Benchmarking and solving dynamic constrained problems, in: IEEE Congress on Evolutionary Computation CEC '09, 2009, pp. 690–697.
[79] Y. Jin, B. Sendhoff, Constructing dynamic optimization test problems using the multi-objective optimization concept, in: G. Raidl, S. Cagnoni, J. Branke, D. Corne, R. Drechsler, Y. Jin, C. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing, Vol. 3005 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 525–536.
[80] J. Mehnen, G. Rudolph, T. Wagner, Evolutionary optimization of dynamic multiobjective functions, Technical Report FI-204/06, Universitat Dortmund, Dortmund, Germany, 2006.
[81] M. Helbig, A. Engelbrecht, Benchmarks for dynamic multi-objective optimisation, in: 2013 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2013, pp. 84–91.
[82] S. Jiang, S. Yang, A framework of scalable dynamic test problems for dynamic multi-objective optimization, in: 2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014, pp. 32–39.
[83] S.Jiang, S. Yang, A benchmark generator for dynamic multi-objective optimization problems, in: 2014 14th UK Workshop on Computational Intelligence (UKCI), 2014, pp. 1–8.
[84] M. Helbig, A. Engelbrecht, Benchmarks for dynamic multi-objective optimisation algorithms, ACM Comput.Sur. 46 (3) (2014) 1–37.
[85] S. Biswas and S. Das and P. N. Suganthan and C. A. C. Coello, Evolutionary multiobjective optimization in dynamic environments: A set of novel benchmark functions, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 3192-3199.
[86] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS'95, 1995, pp. 39–43.
[87] K. Sörensen, Metaheuristicsthe metaphor exposed, Int. Trans. Oper. Res. 22 (1) (2015) 3–18.
[88] Y. Tan, Y. Zhu, Fireworks algorithm for optimization, in: Y. Tan, Y. Shi, K. C. Tan (Eds.), Advances in Swarm Intelligence: First International Conference, ICSI 2010, Beijing, China, June 12–15, 2010, Proceedings, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 355–364.
[89] R. S. Parpinelli, H. S. Lopes, New inspirations in swarm intelligence: a survey, Int. J. Bio-Inspired Comput. 3 (1) (2011) 1–16.
[90] M. Dorigo, T. Stützle, Ant colony optimization, MIT Press, Cambridge, MA, 2004.
[91] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 26 (1) (1996) 29–41.
[92] T. Stützle, H. Hoos, MAX–MIN ant system and local search for the traveling salesman problem, in: IEEE International Conference on Evolutionary Computation, 1997, pp. 309–314.
[93] M. Dorigo, L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[94] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization: An overview, Swarm Intel. 1 (1) (2007) 33–57.
[95] K. E. Parsopoulos, M. N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2010.
[96] Y. Zhang, S. Wang, G. Ji, A comprehensive survey on particle swarm optimization: algorithms and its applications, Math. Probl. Eng. 2015 (2015) 1–38.
[97] T. Blackwell, J. Branke, Multi-swarm optimization in dynamic environments, in: G. Raidl, S. Cagnoni, J. Branke, D. Corne, R. Drechsler, Y. Jin, C. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing, Vol. 3005 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 489–500.
[98] C. Li, S. Yang, A general framework of multipopulation methods with clustering in undetectable dynamic environments, IEEE Trans. Evol. Comput. 16 (4) (2012) 556–577.
[99] S. Yang, C. Li, A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments, IEEE Trans. Evol. Comput. 14 (6) (2010) 959–974.
[100] X. Li, Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, in: K. Deb (Ed.), Genetic and Evolutionary Computation – GECCO 2004, Vol. 3102 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 105–116.
[101] L. Liu, S. Yang, D. Wang, Particle swarm optimization with composite particles in dynamic environments, IEEE Trans. Sys. Man Cybern. Part B: Cybern. 40 (6) (2010) 1634–1648.
[102] D. Parrott, X. Li, A particle swarm model for tracking multiple peaks in a dynamic environment using speciation, in: Proceedings of the 2004 IEEE Congress on Evolutionary Computation CEC2004, vol. 1, 2004, pp. 98–103.
[103] D. Teodorović, Bee colony optimization (BCO), in: C. Lim, L. Jain, S. Dehuri (Eds.), Innovations in Swarm Intelligence, Vol. 248 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2009, pp. 39–60.
[104] S. Bitam, M. Batouche, E.-G. Talbi, A survey on bee colony algorithms, in: Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, 2010, pp. 1–8.
[105] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intel. Rev. 42 (1) (2014) 21–57.
[106] S. Dasgupta, S. Das, A. Abraham, A. Biswas, Adaptive computational chemotaxis in bacterial foraging optimization: an analysis, IEEE Trans. Evol. Comp. 13 (2009) 919–941.
[107] V. Gazi, K. M. Passino, Bacteria foraging optimization, in: Swarm Stability and Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 233–249.
[108] B. Xing, W.-J. Gao, Fish inspired algorithms, in: Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms, Vol. 62 of Intelligent Systems Reference Library, Springer International Publishing, Cham, 2014, pp. 139–155.
[109] M. Garey, D. Johnson, Computer and intractability: A guide to the theory of NP-completeness, Freeman, San Francisco, 1979.
[110] D. Angus, T. Hendtlass, Ant colony optimisation applied to a dynamically changing problem, in: T. Hendtlass, M. Ali (Eds.), Developments in Applied Artificial Intelligence, Vol. 2358 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 618–627.
[111] M. Guntsch, M. Middendorf, H. Schmeck, An ant colony optimization approach to dynamic TSP, in: Proceedings of the 2001 Genetic and Evolutionary Computation Conference, 2001, pp. 860–867.
[112] M. Mavrovouniotis, S. Yang, Memory-based immigrants for ant colony optimization in changing environments, in: C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, J. Merelo, F. Neri, M. Preuss, H. Richter, J. Togelius, G. Yannakakis (Eds.), Applications of Evolutionary Computation, Vol. 6624 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 324–333.
[113] M. Mavrovouniotis, S. Yang, Interactive and non-interactive hybrid immigrants schemes for ant algorithms in dynamic environments, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1542–1549.
[114] L. Melo, F. Pereira, E. Costa, Multi-caste ant colony algorithm for the dynamic traveling salesperson problem, in: M. Tomassini, A. Antonioni, F. Daolio, P. Buesser (Eds.), Adaptive and Natural Computing Algorithms, Vol. 7824 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 179–188.
[115] L. Melo, F. Pereira, E. Costa, Extended experiments with ant colony optimization with heterogeneous ants for large dynamic traveling salesperson problems, in: 2014 14th International Conference on Computational Science and Its Applications (ICCSA), 2014, pp. 171–175.
[116] M. Mavrovouniotis, S. Yang, Ant colony optimization with self-adaptive evaporation rate in dynamic environments, in: 2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014, pp. 47–54.
[117] J. Liu, Rank-based ant colony optimization applied to dynamic traveling salesman problems, Eng. Optim. 37 (8) (2005) 831–847.

[118] U. Boryczka, Ł. Strąk, A hybrid discrete particle swarm optimization with pheromone for dynamic traveling salesman problem, in: N.-T. Nguyen, P. Hoang, K.and Jędrzejowicz (Eds.), Computational Collective Intelligence, Technologies and Applications, Vol. 7654 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 503–512.

[119] U. Boryczka, Ľ. Strąk, Efficient DPSO neighbourhood for dynamic traveling salesman problem, in: C. Bădică, N. Nguyen, M. Brezovan (Eds.), Computational Collective Intelligence, Technologies and Applications, Vol. 8083 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 721–730.

[120] U. Boryczka, Ł. Strąk, Heterogeneous DPSO algorithm for DTSP, in: M. Núñez, T. N. Nguyen, D. Camacho, B. Trawiński (Eds.), Computational Collective Intelligence: 7th International Conference, ICCCI 2015, Madrid, Spain, September 21–23, 2015, Proceedings, Part II, Springer International Publishing, Cham, 2015, pp. 119–128.

[121] U. Boryczka, Ł. Strąk, Diversification and entropy improvement on the DPSO algorithm for DTSP, in: T. N. Nguyen, B. Trawiński, R. Kosala (Eds.), Intelligent Information and Database Systems: 7th Asian Conference, ACIIDS 2015, Bali, Indonesia, March 23–25, 2015, Proceedings, Part I, Springer International Publishing, Cham, 2015, pp. 337–347.

[122] J. Euchi, A. Yassine, H. Chabchoub, The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach, Swarm and Evol. Comput. 21 (2015) 41–53.

[123] B. van Veen, M. Emmerich, Z. Yang, T. Bäck, J. Kok, Ant colony algorithms for the dynamic vehicle routing problem with time windows, in: J. M. Ferrández Vicente, J. R. Álvarez Sánchez, F. de la Paz López, F. J. Toledo Moreo (Eds.), Natural and Artificial Computation in Engineering and Medical Applications: 5th International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2013, Mallorca, Spain, June 10-14, 2013, Proceedings, Part II, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 1–10.

[124] Z. Yang, M. Emmerich, T. Bäck, Ant based solver for dynamic vehicle routing problem with time windows and multiple priorities, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 2813–2819.

[125] M. Ankerl, A. Hämmerle, Applying ant colony optimisation to dynamic pickup and delivery, in: R. Moreno-Díaz, F. Pichler, A. Quesada-Arencibia (Eds.), Computer Aided Systems Theory - EUROCAST 2009, Vol. 5717 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 721–728.

[126] M. Mavrovouniotis, S. Yang, Dynamic vehicle routing: A memetic ant colony optimization approach, in: A. Uyar, E. Ozcan, N. Urquhart (Eds.), Automated Scheduling and Planning, Vol. 505 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2013, pp. 283–301.

[127] S. Gao, Y. Wang, J. Cheng, Y. Inazumi, Z. Tang, Ant colony optimization with clustering for solving the dynamic location routing problem, Appl. Math. Comput. 285 (2016) 149–173.

[128] M. Khouadjia, L. Jourdan, E.-G. Talbi, Adaptive particle swarm for solving the dynamic vehicle routing problem, in: 2010 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), 2010, pp. 1–8.

[129] M. R. Khouadjia, B.Sarasola, E. Alba, L. Jourdan, E.-G. Talbi, A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests, Appl. Soft Comput. 12 (4) (2012) 1426–1439.

[130] M. Khouadjia, E. Alba, L. Jourdan, E.-G. Talbi, Multi-swarm optimization for dynamic combinatorial problems: A case study on dynamic vehicle routing problem, in: M. Dorigo, M. Birattari, G. Di Caro, R. Doursat, A. Engelbrecht, D. Floreano, L. Gambardella, R. Groß, E. Şahin, H. Sayama, T. Stützle (Eds.), Swarm Intelligence, Vol. 6234 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 227–238.

[131] M. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, E.-G. Talbi, Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems, in: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011, pp. 395–403.

[132] M. Khouadjia, E.-G. Talbi, L. Jourdan, E. Sarasola, B.and Alba, Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems, J. Supercomput. 63 (3) (2013) 836–853.

[133] M. Okulewicz, J. Mańdziuk, Application of particle swarm optimization algorithm to dynamic vehicle routing problem, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, J. Zurada (Eds.), Artificial Intelligence and Soft Computing, Vol. 7895 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 547–558.

[134] M. Okulewicz, J. Mańdziuk, Two-phase multi-swarm PSO and the dynamic vehicle routing problem, in: 2014 IEEE Symposium on Computational Intelligence for Human-like Intelligence (CIHLI), 2014, pp. 1–8.

[135] Y. E. Demirtaş, E. Özdemir, U. Demirtaş, A particle swarm optimization for the dynamic vehicle routing problem, in: Modeling, Simulation, and Applied Optimization (ICMSAO), 2015 6th International Conference on, 2015, pp. 1–5.

[136] R. Zhou, H. P. Lee, A. Nee, Applying ant colony optimization algorithm to dynamic job shop scheduling problems, Int. J. Manuf. Res. 3 (3) (2008) 301–320.

[137] A. Baykasoğlu, F. Ozsoydan, An improved firefly algorithm for solving dynamic multidimensional knapsack problems, Expert Sys. Appl. 41 (8) (2014) 3712–3725.

[138] C. Fernandes, A. Rosa, V. Ramos, Binary ant algorithm, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, ACM, New York, NY, 2007, pp. 41–48.

[139] T. Stützle, H. H. Hoos, $\mathcal{MAX}$–$\mathcal{MIN}$ ant system, Future Gen. Comput. Syst. 16 (8) (2000) 889–914.

[140] A. Hashemi, M. Meybodi, A multi-role cellular PSO for dynamic environments, in: Computer Conference, 2009. CSICC 2009. 14th International CSI, 2009, pp. 412–417.

[141] H. Wang, D. Wang, S. Yang, Triggered memory-based swarm optimization in dynamic environments, in: M. Giacobini (Ed.), Applications of Evolutionary Computing, Vol. 4448 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 637–646.

[142] H. Wang, S. Yang, W. Ip, D. Wang, A particle swarm optimization based memetic algorithm for dynamic optimization problems, Nat. Comput. 9 (3) (2010) 703–725.

[143] A. Sharifi, J. K. Kordestani, M. Mahdaviani, M. R. Meybodi, A novel hybrid adaptive collaborative approach based on particle swarm optimization and local search for dynamic optimization problems, Appl. Soft Comput. 32 (2015) 432–448.

[144] H. Wang, S. Yang, W. Ip, D. Wang, A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems, Int. J. Syst. Sci. 43 (7) (2012) 1268–1283.

[145] M. Daneshyari, G. Yen, Dynamic optimization using cultural based PSO, in: 2011 IEEE Congress on Evolutionary Computation (CEC), 2011, pp. 509–516.

[146] T. Blackwell, J. Branke, X. Li, Particle swarms for dynamic optimization problems, in: C. Blum, D. Merkle (Eds.), Swarm Intelligence, Natural Computing Series, Springer Berlin Heidelberg, 2008, pp. 193–217.

[147] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, S. Yang, An adaptive multi-population framework for locating and tracking multiple optima, IEEE Trans. Evol. Comput. 20 (4) (2016) 590–605.

[148] C. Li, S. Yang, M. Yang, An adaptive multi-swarm optimizer for dynamic optimization problems, Evol. Comput. 22 (22) (2014) 559–594.

[149] C. Li, S. Yang, A clustering particle swarm optimizer for dynamic optimization, in: IEEE Congress on Evolutionary Computation. CEC '09, 2009, pp. 439–446.

[150] X. Li, J. Branke, T. Blackwell, Particle swarm with speciation and adaptation in a dynamic environment, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, ACM, New York, NY, 2006, pp. 51–58.

[151] M. Kamosi, A. Hashemi, M. Meybodi, A hibernating multi-swarm optimization algorithm for dynamic environments, in: 2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC), 2010, pp. 363–369.

[152] H. Wang, N. Wang, D. Wang, Multi-swarm optimization algorithm for dynamic optimization problems using forking, in: Control and Decision Conference, 2008. CCDC 2008, Chinese, 2008, pp. 2415–2419.

[153] I. Rezazadeh, M. Meybodi, A. Naebi, Adaptive particle swarm optimization algorithm for dynamic environments, in: Y. Tan, Y. Shi, Y. Chai, G. Wang (Eds.), Advances in Swarm Intelligence, Vol. 6728 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 120–

129.
[154] R. I. Lung, D. Dumitrescu, Evolutionary swarm cooperative optimization in dynamic environments, Nat.Comput. 9 (1) (2010) 83–94.
[155] J. Kordestani, A. Rezvanian, M. Meybodi, CDEPSO: a bi-population hybrid approach for dynamic optimization problems, Appl. Intell. 40 (4) (2014) 682–694.
[156] A. Hashemi, M. Meybodi, Cellular PSO: A PSO for dynamic environments, in: Z. Cai, Z. Li, Z. Kang, Y. Liu (Eds.), Advances in Computation and Intelligence, Vol. 5821 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 422–433.
[157] R. Lung, D. Dumitrescu, A collaborative model for tracking optima in dynamic environments, in: IEEE Congress on Evolutionary Computation CEC 2007, 2007, pp. 564–567.
[158] M. Kamosi, A. Hashemi, M. Meybodi, A new particle swarm optimization algorithm for dynamic environments, in: B. Panigrahi, S. Das, P. Suganthan, S. Dash (Eds.), Swarm, Evolutionary, and Memetic Computing, Vol. 6466 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 129–138.
[159] C. Li, S. Yang, Fast multi-swarm optimization for dynamic optimization problems, in: Fourth International Conference on Natural Computation, ICNC '08, vol. 7, 2008, pp. 624–628.
[160] S. Esquivel, C. Coello, Particle swarm optimization in non-stationary environments, in: C. Lemaître, C. Reyes, J. Gonzlez (Eds.), Advances in Artificial Intelligence-IBERAMIA 2004, Vol. 3315 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 757–766.
[161] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inf. Sci.178 (15) (2008) 3096–3109
[162] S. Janson, M. Middendorf, A hierarchical particle swarm optimizer for dynamic optimization problems, in: G. Raidl, S. Cagnoni, J. Branke, D. Corne, R. Drechsler, Y. Jin, C. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing, Vol. 3005 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 513–524.
[163] T. Blackwell, Particle swarm optimization in dynamic environments, in: Evolutionary Computation in Dynamic and Uncertain Environments, Vol. 51 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2007, pp. 29–49.
[164] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Trans. Evol. Comput. 10 (4) (2006) 459–472.
[165] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Trans. Evol. Comput. 10 (4) (2006) 440–458.
[166] L. Liu, D. Wang, S. Yang, Compound particle swarm optimization in dynamic environments, in: M. Giacobini, A. Brabazon, S. Cagnoni, G. Di Caro, R. Drechsler, A. Ekárt, A. Esparcia-Alcázar, M. Farooq, A. Fink, J. McCormack, M. ONeill, J. Romero, F. Rothlauf, G. Squillero, A. Uyar, S. Yang (Eds.), Applications of Evolutionary Computing, Vol. 4974 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 617–626.
[167] L. Liu, D. Wang, J. Tang, Composite particle optimization with hyper-reflection scheme in dynamic environments, Appl. Soft Comput. 11 (8) (2011) 4626–4639.
[168] W. Luo, J. Sun, C. Bu, H. Liang, Species-based particle swarm optimizer enhanced by memory for dynamic optimization, Appl. Soft Comput. 47 (2016) 130–140.
[169] H. Chen, L. Ma, M. He, X. Wang, X. Liang, L. Sun, M. Huang, Artificial bee colony optimizer based on bee life-cycle for stationary and dynamic optimization, IEEE Trans. Sys. Man Cyber.: Sys. PP (99) (2016) 1–20. doi:10.1109/TSMC.2016.2558045.
[170] S. K. Nseef, S. Abdullah, A. Turky, G. Kendall, An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems, Knowl.-Based Sys. 104 (2016) 14 – 23.
[171] H. Nakano, M. Kojima, A. Miyauchi, An artificial bee colony algorithm with a memory scheme for dynamic optimization problems, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 2657–2663.
[172] D. Yazdani, A. Sepas-Moghaddam, A. Dehban, N. Horta, A novel approach for optimization in dynamic environments based on modified artificial fish swarm algorithm, Int. J. of Comput. Intell. Appl. 15 (02) (2016) 1650010.
[173] W. Tang, Q. Wu, J. Saunders, Bacterial foraging algorithm for dynamic environments, in: IEEE Congress on Evolutionary Computation CEC 2006, 2006, pp. 1324–1330.
[174] J. Abbott, A. Engelbrecht, Performance of bacterial foraging optimization in dynamic environments, in: M. Dorigo, M. Birattari, C. Blum, A. Christensen, A. Engelbrecht, R. Groß, T. Stützle (Eds.), Swarm Intelligence, Vol. 7461 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 284–291.
[175] M. Daas, M. Batouche, Multi-bacterial foraging optimization for dynamic environments, in: Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of, 2014, pp. 237–242.
[176] D. Yazdani, M. Akbarzadeh-Totonchi, B. Nasiri, M. Meybodi, A new artificial fish swarm algorithm for dynamic optimization problems, in: 2012 IEEE Congress on Evolutionary Computation (CEC), 2012, pp. 1–8.
[177] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M.Meybodi, M. Akbarzadeh-Totonchi, mNAFSA: A novel approach for optimization in dynamic environments with global changes, Swarm and Evol. Comput. 18 (2014) 38–53.
[178] A. Abshouri, M. Meybodi, A. Bakhtiary, New firefly algorithm based on multi swarm and learning automata in dynamic environments, in: IEEE Proceedings, 2011, pp. 73–77.
[179] B. Nasiri, M. Meybodi, Speciation based firefly algorithm for optimization in dynamic environments, Int. J. Artif. Intel. 8 (12) (2012) 118–132.
[180] B. Nasiri, M. R. Meybodi, History-driven firefly algorithm for optimisation in dynamic and uncertain environments, Int. J. Bio-Inspired Comput. 8 (5) (2016) 326–339.
[181] B. Nasiri, M. Meybodi, M. Ebadzadeh, History-driven particle swarm optimization in dynamic and uncertain environments, Neurocomputing 172 (2016) 356 – 370.
[182] F. B. Ozsoydan, A. Baykasoglu, A multi-population firefly algorithm for dynamic optimization problems, in: Evolving and Adaptive Intelligent Systems (EAIS), 2015 IEEE International Conference on, 2015, pp. 1–7.
[183] S. Farahani, B. Nasiri, M. Meybodi, A multiswarm based firefly algorithm in dynamic environments, in: The Third International Conference on Signal Processing Systems (ICSPS 2011), Vol. 3, 2011, pp. 68–72.
[184] P. Korosec, J. Silc, The continuous differential ant-stigmergy algorithm applied to dynamic optimization problems, in: 2012 IEEE Congress on Evolutionary Computation (CEC), 2012, pp. 1–8.
[185] P. Korosec, J. Silc, The differential ant-stigmergy algorithm applied to dynamic optimization problems, in: IEEE Congress on Evolutionary Computation. CEC '09, 2009, pp. 407–414.
[186] J. Fernandez-Marquez, J. Arcos, Adapting particle swarm optimization in dynamic and noisy environments, in: 2010 IEEE Congress on Evolutionary Computation (CEC), 2010, pp. 1–8.
[187] I. Rezazadeh, M. Meybodi, A. Naebi, Particle swarm optimization algorithm in dynamic environments: Adapting inertia weight and clustering particles, in: 2011 Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS), 2011, pp. 76–82.
[188] D. Bose, S. Biswas, S. Kundu, S. Das, A strategy pool adaptive artificial bee colony algorithm for dynamic environment through multi-population approach, in: S. Panigrahi, B.and Das, P. Suganthan, P.N.and Nanda (Eds.), Swarm, Evolutionary, and Memetic Computing, Vol. 7677 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 611–619.
[189] S. Biswas, D. Bose, S. Kundu, A clustering particle based artificial bee colony algorithm for dynamic environment, in: B. Panigrahi, S. Das, P. Suganthan, P. Nanda (Eds.), Swarm, Evolutionary, and Memetic Computing, Vol. 7677 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 151–159.
[190] W. Tfaili, P. Siarry, A new charged ant colony algorithm for continuous dynamic optimization, Appl. Math. Comput. 197 (2), 2008, 604 – 613.
[191] J. Dréo, P. Siarry, An ant colony algorithm aimed at dynamic continuous optimization, Appl. Math. Comput. 181 (1) (2006) 457 – 467.
[192] X. Zheng, H. Liu, A different topology multi-swarm PSO in dynamic environment, in: IEEE International Symposium on IT in Medicine Education, 2009 ITIME '09, vol. 1, 2009, pp. 790–795.
[193] K. Parsopoulos, M. Vrahatis, Unified particle swarm optimization in dynamic environments, in: F. Rothlauf, J. Branke, S. Cagnoni, D. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing, Vol. 3449

of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 590–599.

[194] Y. Jiang, W. Huang, L. Chen, Applying multi-swarm accelerating particle swarm optimization to dynamic continuous functions, in: Second International Workshop on Knowledge Discovery and Data Mining, WKDD 2009, 2009, pp. 710 –713.

[195] R. Takano, T. Harada, H. Sato, K. Takadama, Artificial bee colony algorithm based on local information sharing in dynamic environment, in: H. Handa, H. Ishibuchi, Y.-S. Ong, K. Tan (Eds.), Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, Vol. 1 of Proceedings in Adaptation, Learning and Optimization, Springer International Publishing, Cham, 2015, pp. 627–641.

[196] M. Kojima, H. Nakano, A. Miyauchi, An artificial bee colony algorithm for solving dynamic optimization problems, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 2398–2405.

[197] S. Raziuddin, S. Sattar, R. Lakshmi, M. Parvez, Differential artificial bee colony for dynamic environment, in: N. Meghanathan, B. Kaushik, D. Nagamalai (Eds.), Advances in Computer Science and Information Technology, Vol. 131 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2011, pp. 59–69.

[198] R. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of the 2001 IEEE Congress on Evolutionary Computation, vol. 1, 2001, pp. 94–100.

[199] X. Hu, R. Eberhart, Adaptive particle swarm optimization: detection and response to dynamic systems, in: Proceedings of the 2002 IEEE Congress on Evolutionary Computation. CEC '02, vol. 2, 2002, pp. 1666–1670.

[200] C. Li, T. T. Nguyen, M. Yang, S. Yang, S. Zeng, Multi-population methods in unconstrained continuous dynamic environments: The challenges, Inf. Sci. 296 (2015) 95 – 118.

[201] R. Thomsen, Multimodal optimization using crowding-based differential evolution, in: Proceedings of the 2004 IEEE Congress on Evolutionary Computation CEC2004, vol. 2, 2004, pp. 1382–1389.

[202] H. Dewan, R. Nayak, Solving dynamic constraint single objective functions using a nature inspired technique, in: Information Science and Applications (ICISA), 2014 International Conference on, 2014, pp. 1–6.

[203] Y. Yin, L. Sun, Generalized dynamic constraint satisfaction based on extension particle swarm optimization algorithm for collaborative simulation, in: 2007 10th IEEE International Conference on Computer-Aided Design and Computer Graphics, 2007, pp. 541–544.

[204] J. Wei, Y. Wang, Hyper rectangle search based particle swarm algorithm for dynamic constrained multi-objective optimization problems, in: 2012 IEEE Congress on Evolutionary Computation (CEC), 2012, pp. 1–8.

[205] J. Wei, L. Jia, A novel particle swarm optimization algorithm with local search for dynamic constrained multi-objective optimization problems, in: 2013 IEEE Congress on Evolutionary Computation (CEC), 2013, pp. 2436–2443.

[206] M. Greeff, A. Engelbrecht, Dynamic multi-objective optimisation using PSO, in: N. Nedjah, L. d. S. Coelho, L. d. M. Mourelle (Eds.), Multi-Objective Swarm Intelligent Systems, Vol. 261 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2010, pp. 105–123.

[207] M. Helbig, A. Engelbrecht, Dynamic multi-objective optimization using PSO, in: E. Alba, A. Nakib, P. Siarry (Eds.), Metaheuristics for Dynamic Optimization, Vol. 433 of Studies in Computational Intelligence, Springer Berlin Heidelberg, 2013, pp. 147–188.

[208] A. Rakitianskaia, A. Engelbrecht, Training feedforward neural networks with dynamic particle swarm optimisation, Swarm Intel. 6 (3) (2012) 233–270.

[209] A. Rakitianskaia, A. P. Engelbrecht, Training neural networks with PSO in dynamic environments, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 667–673.

[210] Z. Michalewicz, Quo vadis, evolutionary computation?, in: J. Liu, C. Alippi, B. Bouchon-Meunier, G. W. Greenwood, H. A. Abbass (Eds.), Advances in Computational Intelligence: IEEE World Congress on Computational Intelligence, WCCI 2012, Brisbane, Australia, June 10–15, 2012. Plenary/Invited Lectures, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 98–121.

[211] G. D. Caro, M. Dorigo, Ant colonies for adaptive routing in packet-switched communications networks, in: A. E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature – PPSN V, Vol. 1498 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 673–682.

[212] G. D. Caro, M. Dorigo, AntNet: Distributed stigmergetic control for communications networks, J. Artif. Intel. Res. 9 (1998) 317–365.

[213] D. Subramanian, P. Druschel, J. Chen, Ants and reinforcement learning: A case study in routing in dynamic networks, in: Proceedings of International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1997, pp. 832–838.

[214] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, Ant-based load balancing in telecommunications networks, Adapt. Behav. 5 (2) (1996) 169–207.

[215] O. Verma, M. Sharma, N. Gupta, P. Nanda, S. Chawla, A new approach to dynamic network routing using omicron ant colony algorithm, in: 2011 3rd International Conference on Electronics Computer Technology (ICECT), vol. 5, 2011, pp. 177–181.

[216] K. Oida, M. Sekido, An agent-based routing system for qos guarantees, in: IEEE SMC'99 Conference Proceedings, 1999 IEEE International Conference on Systems, Man, and Cybernetics, 1999, vol. 3, 1999, pp. 833–838.

[217] H. Sandalidis, K. Mavromoustakis, P. Stavroulakis, Performance measures of an ant based decentralized routing scheme for circuit switching communication networks, Soft Comput. 5 (4) (2001) 313–317.

[218] H. Sandalidis, K. Mavromoustakis, P. Stavroulakis, Ant-based probabilistic routing with pheromone and antipheromone mechanisms, Int. J. Commun. Syst. 17 (2004) 55–62.

[219] K. Sim, W. Sun, Ant colony optimization for routing and loadbbalancing: Survey and new directions, IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans 33 (5) (2003) 560–572.

[220] T. White, B. Pagurek, F. Oppacher, Connection management using adaptive mobile agents, in: H. Arabnia (Ed.), Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, CSREA Press, Boston, MA, 1998, pp. 802–809.

[221] D. Câmara, A. Loureiro, GPS/Ant-like routing in ad hoc networks, Telecommun.Syst. 18 (1-3) (2001) 85–100.

[222] G. D. Caro, F. Ducatelle, L. M. Gambardella, Anthocnet: An ant-based hybrid routing algorithm for mobile ad hoc networks, in: X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervós, J. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII, Vol. 3242 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 461–470.

[223] G. D. Caro, F. Ducatelle, L. M. Gambardella, Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks, Eur. Trans. on Telecommun. 16 (5) (2005) 443–455.

[224] M. Gunes, U. Sorges, I. Bouazizi, ARA–The ant-colony based routing algorithm for MANETS, in: Proceedings of the International Conference on Parallel Processing Workshops on Ad Hoc Networks, 2002, pp. 79–85.

[225] G. Singh, N. Kumar, A. Verma, Antalg: An innovative ACO based routing algorithm for MANETs, J. Netw. Comput. Appl. 45 (2014) 151–167.

[226] C.-B. Liu, H.-J. Wang, Z.-P. Luo, X.-Q. Yu, L.-H. Liu, Qos multicast routing problem based on artificial fish-swarm algorithm, in: First International Workshop on Education Technology and Computer Science. ETCS '09, vol. 2, 2009, pp. 814–817.

[227] H. Wedde, M. Farooq, T. Pannenbaecker, B. Vogel, C. Mueller, J. Meth, R. Jeruschkat, BeeAdHoc: An energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO '05, ACM, New York, NY, 2005, pp. 153–160.

[228] F. Barani, A. Barani, Dynamic intrusion detection in AODV-based MANETs using memetic artificial bee colony algorithm, in: 2014 22nd Iranian Conference on Electrical Engineering (ICEE), 2014, pp. 1040–1046.

[229] H. Dewan, R. Nayak, V. Devi, A peer-to-peer dynamic single objective particle swarm optimizer, in: B. Panigrahi, P. Suganthan, S. Das, S. Dash (Eds.), Swarm, Evolutionary, and Memetic Computing, Vol. 8297 of Lecture Notes in Computer Science, Springer International Publishing, 2013, pp. 630–641.

[230] H. Dewan, R. Nayak, V. Devi, A peer-to-peer dynamic multi-objective particle swarm optimizer, in: R. Prasath, T. Kathirvalavakumar (Eds.), Mining Intelligence and Knowledge Exploration, Vol. 8284 of Lecture Notes in Computer Science, Springer International Publishing, 2013, pp.

453–465.

[231] A. Vogel, M. Fischer, H. Jaehn, T. Teich, Real-world shop floor scheduling by ant colony optimization, in: M. Dorigo, G. D. Caro, M. Sampels (Eds.), Ant Algorithms, Vol. 2463 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 268–273.

[232] S. Nakrani, C. Tovey, On honey bees and dynamic server allocation in internet hosting centers, Adapt. Behav. Animals, Animats, Software Agents, Robots, Adaptive Systems 12 (3-4) (2004) 223–240.

[233] P. Triwate, P. Luangpaiboon, Bees algorithm for dynamic multi-zone dispatching in truck load trucking, in: 2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2010, pp. 1165–1169.

[234] H. Wedde, S. Lehnhoff, B. van Bonn, Z. Bay, S. Becker, S. Bottcher, C. Brunner, A. Buscher, T. Furst, A. Lazarescu, E. Rotaru, S. Senge, B. Steinbach, F. Yilmaz, T. Zimmermann, A novel class of multi-agent algorithms for highly dynamic transport planning inspired by honey bee behavior, in: IEEE Conference on Emerging Technologies and Factory Automation, ETFA, 2007, pp. 1157–1164.

[235] M. Sulaiman, M. Mustafa, A. Azmi, O. Aliman, S. Abdul Rahim, Optimal allocation and sizing of distributed generation in distribution system via firefly algorithm, in: Power Engineering and Optimization Conference (PEDCO) Melaka, Malaysia, 2012 IEEE International, 2012, pp. 84–89.

[236] D. Teodorović, M. Dell'Orco, Bee colony optimization - a cooperative learning approach to complex transportation problems, in: Advanced OR and AI Methods in Transportation: Proceedings of 16th MiniEURO Conference and 10th Meeting of EWGT, 2005, pp. 51–60.

[237] M. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, P.Melin, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, Appl. Soft Comput. 9 (3) (2009) 1102–1110.

[238] M. K. Rath, B. B. V. L. Deepak, PSO based system architecture for path planning of mobile robot in dynamic environment, in: 2015 Global Conference on Communication Technologies (GCCT), 2015, pp. 797–801.

[239] W. Li, J. Lei, T. Wang, C. Xiong, J. Wei, Dynamic optimization for resource allocation in relay-aided OFDMA systems under multiservice, IEEE Trans. Vehic. Technol. 65 (3) (2016) 1303–1313.

[240] H.-P. Hsu, A HPSO for solving dynamic and discrete berth allocation problem and dynamic quay crane assignment problem simultaneously, Swarm and Evol. Comput. 27 (2016) 156 – 168.

[241] J. Eaton, S. Yang, M. Mavrovouniotis, Ant colony optimization with immigrants schemes for the dynamic railway junction rescheduling problem with multiple delays, Soft Comput. 20 (8) (2016) 2951–2966.

[242] K. Vaisakh, P. Praveena, S. R. M. Rao, K. Meah, Solving dynamic economic dispatch problem with security constraints using bacterial foraging PSO-DE algorithm, Int. J. Electr. Power Energy Syst. 39 (1) (2012) 56 – 67.

[243] B. Panigrahi, V. R. Pandi, S. Das, Adaptive particle swarm optimization approach for static and dynamic economic load dispatch, Energy Conv. Manag. 49 (6) (2008) 1407–1415.

[244] Y. Wang, J. Zhou, Y. Lu, H. Qin, Y. Wang, Chaotic self-adaptive particle swarm optimization algorithm for dynamic economic dispatch problem with valve-point effects, Expert Syst. Appl. 38 (11) (2011) 14231–14237.

[245] X. Yuan, A. Su, Y. Yuan, H. Nie, L. Wang, An improved PSO for dynamic load dispatch of generators with valve-point effects, Energy 34 (1) (2009) 67–74.

[246] T. A. A. Victoire, A. E. Jeyakumar, Deterministically guided PSO for dynamic dispatch considering valve-point effect, Electr. Power Syst. Res. 73 (3) (2005) 313–322.

[247] T. Niknam, R. Azizipanah-Abarghooee, A. Roosta, Reserve constrained dynamic economic dispatch: a new fast self-adaptive modified firefly algorithm, IEEE Syst. J. 6 (4) (2012) 635–646.

[248] P. Prudhvi, A. Sudarshan, C. Bezawada, A improved artificial fish swarming optimization for economic load dispatch with dynamic constraints, in: K. Deep, A. Nagar, M. Pant, J. Bansal (Eds.), Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011), Vol. 130 of Advances in Intelligent and Soft Computing, Springer India, 2012, pp. 141–149.

[249] F. Abu-Mouti, M. El-Hawary, Optimal dynamic economic dispatch including renewable energy source using artificial bee colony algorithm, in: Systems Conference (SysCon), 2012 IEEE International, 2012, pp. 1–6.

[250] T. Niknam, F. Golestaneh, Enhanced bee swarm optimization algorithm for dynamic economic dispatch, IEEE Sys. J. 7 (4) (2013) 754–762.

[251] L. Zhang, G.-X. Wang, J. Zhu, X.-G. He, P. Liu, Artificial bee colony algorithm for optimal dynamics dispatch problem, in: 2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), vol. 1, 2012, pp. 314–317.

[252] P. Lu, J. Zhou, H. Zhang, R. Zhang, C. Wang, Chaotic differential bee colony optimization algorithm for dynamic economic dispatch problem with valve-point effects, Int. J. of Electr. Power Energy Syst. 62 (2014) 130–143.

[253] S. Hemamalini, S. Simon, Dynamic economic dispatch using artificial bee colony algorithm for units with valve-point effect, Eur. Trans. Electr. Power 21 (1) (2011) 70–81.

[254] Y. Chen, J. Wen, A hybrid intelligent optimization algorithm for dynamic economic dispatch with valve-point effects, in: Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific, 2012, pp. 1–5.

[255] T. Niknam, M. Narimani, J. Aghaei, S. Tabatabaei, M. Nayeripour, Modified honey bee mating optimisation to solve dynamic optimal power flow considering generator constraints, Gen. Transm. Distrib. IET 5 (10) (2011) 989–1002.

[256] W. Tang, M. Li, S. He, Q. Wu, J. Saunders, Optimal power flow with dynamic loads using bacterial foraging algorithm, in: International Conference on Power System Technology PowerCon 2006, 2006, pp. 1–5.

[257] W. Tang, M. Li, Q. Wu, J. Saunders, Bacterial foraging algorithm for optimal power flow in dynamic environments, IEEE Trans. Circuits Syst. I: Reg. Pap. 55 (8) (2008) 2433–2442.

[258] R. Takano, D. Yamazaki, Y. Ichikawa, K. Hattori, K. Takadama, Multiagent-based ABC algorithm for autonomous rescue agent cooperation, in: 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2014, pp. 585–590.

[259] O. Bendjeghaba, Continuous firefly algorithm for optimal tuing of PID controller in AVR system, J. of Electr. Eng.65 (1) (2014) 44–49.

[260] Z.-L. Gaing, A particle swarm optimization approach for optimum design of PID controller in AVR system, IEEE Trans. on Energy Conv. 19 (2) (2004) 384–391.

[261] D. Kim, J. Cho, A biologically inspired intelligent PID controller tuning for AVR systems, Int. J. Control Autom. Syst. 4 (5) (2006) 624–636.

[262] H. M. Gomes, Truss optimization with dynamic constraints using a particle swarm algorithm, Expert Syst. Appl. 38 (1) (2011) 957 – 968.

[263] H. Gomes, A firefly metaheuristic structural size and shape optimisation with natural frequency constraints, Int. J. Metaheuristics 2 (1) (2012) 38–55.

[264] V. Sesum-Cavic, E. Kuhn, Comparing configurable parameters of swarm intelligence algorithms for dynamic load balancing, in: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010, pp. 42–49.

[265] M. Hossain, I. Ferdous, Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique, Robot. Auton. Syst. 64 (2015) 137–141.

[266] A. Nasrollahy, H. Javadi, Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target, in: Third UKSim European Symposium on Computer Modeling and Simulation EMS '09, 2009, pp. 60–65.

[267] H.-Q. Min, J.-H. Zhu, X.-J. Zheng, Obstacle avoidance with multi-objective optimization by PSO in dynamic environment, in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 5, 2005, pp. 2950–2956.

[268] R. Palm, A. Bouguerra, Particle swarm optimization of potential fields for obstacle avoidance, in: Recent Advances in Robotics and Mechatronics, 2013, pp. 117–123.

[269] L. Liu, S. Ranjithan, G. Mahinthakumar, Contamination source identification in water distribution systems using an adaptive dynamic optimization procedure, J. Water Resour. Plann. Manag. 137 (2) (2011) 183–192.