IMA Journal of Management Mathematics (2013) **24**, 451–480
doi:10.1093/imaman/dps021
Advance Access publication on 10 October 2012

Metaheuristics for dynamic combinatorial optimization problems

Shengxiang Yang*
College of Math and Statistics, Nanjing University of Information Science and Technology, Nanjing 210044, China, and School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK
*Corresponding author: syang@dmu.ac.uk

Yong Jiang
College of Math and Physics, Nanjing University of Information Science and Technology, Nanjing 210044, China
jiang@nuist.edu.cn

and

Trung Thanh Nguyen
School of Engineering, Technology and Maritime Operations, Liverpool John Moores University, Liverpool L3 3AF, UK
T.T.Nguyen@ljmu.ac.uk

[Received on 4 April 2012; accepted on 21 June 2012]

Many real-world optimization problems are combinatorial optimization problems subject to dynamic environments. In such dynamic combinatorial optimization problems (DCOPs), the objective, decision variables and/or constraints may change over time, and so solving DCOPs is a challenging task. Metaheuristics are a good choice of tools to tackle DCOPs because many metaheuristics are inspired by natural or biological evolution processes, which are always subject to changing environments. In recent years, DCOPs have attracted a growing interest from the metaheuristics community. This paper is a tutorial on metaheuristics for DCOPs. We cover the definition of DCOPs, typical benchmark problems and their characteristics, methodologies and performance measures, real-world case study and key challenges in the area. Some future research directions are also pointed out in this paper.

Keywords: metaheuristics; genetic algorithm; ant colony optimization; dynamic optimization problem; dynamic combinatorial optimization problem.

1. Introduction

In our daily life, we face various optimization problems, of which many are combinatorial optimization problems (COPs). A COP has a discrete search space (i.e. a binary or combinatorial space instead of a real space), which consists of a finite set of solutions. To solve a COP, an optimization algorithm attempts to find the optimal solution from a large finite set of solutions in the search space. The search space of candidate solutions usually grows exponentially as the size of the problem increases, which makes the search for the optimal solution by a traditional deterministic method (e.g. an exhaustive search or branch-and-bound method) infeasible. For example, the travelling salesman problem (TSP) is a typical example of COP, where the search space of candidate solutions grows even more than exponentially as the number of cities increases. Hence, researchers turn to stochastic optimization methods or heuristics and metaheuristics to solve COPs.

In computer science, a heuristic is a technique designed to solve a given problem, which ignores whether the final solution can be proved to be optimal or not, and usually produces a sufficiently good quality solution. Heuristics are intended to gain computational performance or conceptual simplicity, potentially at the cost of accuracy or precision. A metaheuristic method is a heuristic for solving a general class of computational problems by combining black-box procedures—usually heuristics themselves—in a hopefully efficient way. Over the past several decades, researchers have developed many metaheuristics, which include genetic algorithm (GA) proposed by Holland (1975), simulated annealing (SA) by Kirkpatrick *et al.* (1983), tabu search (TS) by Glover (1986), ant colony optimization (ACO) by Dorigo (1992), artificial immune system (AIS) by Farmer *et al.* (1986) and many more. Metaheuristics have frequently been used for solving many COPs with promising results.

However, in the real world, many optimization problems, including COPs, are subject to dynamic environments, where the optimization objective, decision variables and/or constraints may change over time. Hence, addressing dynamic optimization problems (DOPs) is an important task. DOPs pose serious challenges to deterministic optimization methods as well as metaheuristics due to the characteristics of changing conditions. However, metaheuristics, when properly modified, are a good choice of tools to tackle DOPs because most metaheuristics are inspired by natural or biological evolution processes which are subject to changing environments. Over the past two decades, researchers have developed different methods to modify metaheuristics to solve DOPs, including dynamic COPs (DCOPs). A comprehensive survey on metaheuristics, especially evolutionary algorithms (EAs), for general DOPs can be found in Jin & Branke (2005), Cruz *et al.* (2011) and Nguyen *et al.* (2012). There are also several monographs (see Branke, 2001; Morrison, 2004; Weicker, 2003), edited books (see Yang *et al.*, 2007) and PhD theses (see Li, 2011; Liekens, 2005; Nguyen, 2011) that deal with this area of research.

In recent years, DCOPs have attracted a growing interest from the metaheuristics community with some promising results. In this paper, we provide a tutorial on metaheuristics for DCOPs. We first introduce the definition of DCOPs, describe typical benchmark problems and summarize the basic features of DCOPs in the literature. Then, we briefly review the methodologies developed to modify metaheuristics to deal with DCOPs and relevant performance measures to evaluate metaheuristics for DCOPs. Thereafter, we present a case study on metaheuristics for real-world DCOPs. Finally, we discuss some key issues and challenges in the domain. Some future research directions regarding metaheuristics for DCOPs are also discussed in this paper.

The rest of this paper is organized as follows. Section 2 presents the definition, brief review, typical benchmark problems, and characteristics of DCOPs studied in the literature. Section 3 overviews the development of metaheuristics for DCOPs, including the performance measures and methodologies developed for modifying metaheuristics for solving DCOPs. Section 4 presents a real-world case study. Section 5 gives discussions on some key challenges on metaheuristics for DCOPs. Finally, Section 6 concludes this paper with suggestions on the future research directions in the area.

2. Dynamic combinatorial optimization problems

2.1 *Problem definition*

Even defining DCOPs is a challenging task. In general terms, researchers usually define ‘optimization problems that change over time’ as *dynamic problems*, *time-dependent problems* or *DOPs*, which are not distinguished or are used interchangeably. In many existing metaheuristics studies, the considered problem is either defined as a sequence of static problems linked up by some dynamic rules (e.g. Weicker, 2000, 2003; Aragon & Esquivel, 2004; Rohlfshagen & Yao, 2010), or as a problem that have time-dependent parameters in its mathematical model (e.g. Bäck, 1998; Bosman, 2007; Woldesenbet & Yen, 2009) without explicitly mentioning whether the problem is solved online or not, although the authors do imply that the problem is solved online by the algorithm as time goes by. However, it would be better to distinguish a DOP from a general time-dependent problem because, no matter how the problem changes, from the perspective of an optimization algorithm, a time-dependent problem is different from a static problem only if it is solved in a dynamic way, i.e. the algorithm needs to take into account changes during the optimization process as time goes by. In this paper, we define *DOPs* as *a special class of dynamic problems that are solved online by an optimization algorithm as time goes by*. A more formal and detailed definition of DOPs, which considers the common characteristics of DOPs, can be found in Nguyen & Yao (2009) and Nguyen (2011, Chapter 4).

For DCOPs, Rohlfshagen & Yao (2008, 2011) provided a definition that extends the definition of stationary COPs by Garey & Johnson (1979). According to Garey & Johnson (1979), a COP Π has the following three parts: a set S of instances, a finite set $X(I)$ of candidate solutions for each instance $I \in S$ and a function $f : S \times X \to \mathbb{R}$ that assigns a positive rational number $f(I, \vec{x})$ to each instance $I \in S$ and each candidate solution $\vec{x} \in X(I)$. From this COP definition, a DCOP $\Pi_D = (\Pi, \mathcal{T})$ can be defined as

$$\text{optimize}_{\vec{x}(t)} f(I(k), \vec{x}(t)) \quad \text{subject to } \vec{x}(t) \in X(I(k)) \text{ and } I(k+1) = \mathcal{T}(I(k), t), \tag{2.1}$$

where t is the time, k is the index of an environment such that $k \times \tau \leqslant t < (k+1) \times \tau$ and τ is the period of an environment, $\vec{x}(t)$ is a candidate solution at time t for instance $I(k) \in S$, and the mapping $\mathcal{T} : S \times \mathbb{N} \to S$ defines the trajectory through the space of all instances over time (i.e. the state space), and sometimes is called the *meta-dynamics*. The actual time t is usually assumed to be discrete and is measured in the unit of a single function evaluation.

The above definition of DCOP highlights the role of the actual dynamics and the parameters so that a DCOP Π_D is specified not only by S, but also by the set of instances determined by $\mathcal{T}$. Given the above definition, a DCOP can be seen as a discrete dynamical system and the task of an algorithm is to track the trajectory of successive global optima in the search space.

2.2 *DCOPs in the literature: brief review*

In order to compare the performance of metaheuristics for DCOPs, researchers have developed a number of benchmarks or modelled real-world DCOPs. Generally speaking, a DCOP is constructed via changing (the parameters of) a stationary base COP regarding the characteristics of the environmental dynamics, such as the frequency, severity, predictability and cyclicity of the environmental changes. Below, we briefly review the DCOPs that have been used/developed by researchers to test metaheuristics in the literature.

In the early days, the dynamic test environments were quite simple: just switching between two or more stationary problems or between two or more states of one problem. For example, the dynamic knapsack problem (DKP) where the knapsack capacity oscillates between two or more fixed values has been frequently used in the literature (see Lewis *et al.*, 1998; Mori *et al.*, 1997; Ng & Wong, 1995). The DKP has later been extended to dynamic multi-dimensional knapsack problems (DMKP) in several studies (see Branke *et al.*, 2006; Uyar & Uyar, 2009). The dynamic bit-matching problem (DBMP) is another simple DCOP that has been used by researchers for analysing the performance of metaheuristics (see Stanhope & Daida, 1998).

Later, Yang (2004) proposed a DCOP generator based on the concept of problem difficulty and unitation and trap functions. An XOR DOP generator, which can generate dynamic environments from any binary encoded stationary problem based on a bit-wise exclusive-or (XOR) operator, has been proposed by Yang (2003) and Yang & Yao (2005, 2008). Inspired by and based on the XOR DOP generator, Rohlfshagen & Yao (2010) further developed the dynamic modular function that contains decomposable modules with tunable degrees of inter-modular dependencies and hence can be used to test the role of modularity on EAs for DCOPs. Richter (2004, 2010) constructed spatio-temporal fitness landscapes based on the concept of coupled map lattices (CML) (Chazottes & Fernandez, 2005), where properties such as modality, ruggedness, epistasis, dynamic severity and Lyapunov exponents can be defined and analysed. Bosman (2007) and Nguyen & Yao (2009) investigated the online DCOPs and DOPs that have the time-linkage property, i.e. a solution found at a given time by an optimization algorithm affects the future behaviour of the problem. In order to develop a unified approach of constructing DOPs across the discrete and real spaces, a generalized dynamic benchmark generator was recently proposed by Li & Yang (2008) and Li *et al.* (2008), which can be instantiated to construct dynamic test environments for any type of solution space.

In recent years, researchers have also studied a number of real-world DCOPs. For example, Li *et al.* (2006) studied the dynamic TSP (DTSP) where the cities may change their locations. Handa *et al.* (2007) applied a memetic algorithm to solve the dynamic salting truck routing problem based on the real-world data from the South Gloucestershire Council in England and achieved a 10% improvement over the available solutions in terms of the distances travelled by the trucks. Cremers *et al.* (2010) studied a dynamic planning problem for the transport of elderly and disabled people using a GA. Mavrovouniotis & Yang (2010, 2011b,c,d) investigated the DTSP where cities may join or leave the topology over time or the traffic may change over time. The dynamic vehicle routing problems in logistics and transport networks have also been investigated by the metaheuristics community (see Chitty & Hernandez, 2004; Mei *et al.*, 2010; Weise *et al.*, 2009). Cheng & Yang (2010b) and Yang *et al.* (2010) studied the dynamic shortest path routing problem (DSPRP) in mobile *ad hoc* networks (MANETs). Cheng & Yang have also studied the dynamic multicast routing problem in MANETs (see Cheng & Yang, 2010a) and the dynamic load balanced clustering problem in MANETs (see Cheng & Yang, 2011).

2.3 *Typical benchmark problems*

2.3.1 *The dynamic bit-matching problem.* The DBMP has been used by researchers for the analysis of the performance of EAs in dynamic environments (see Droste, 2002; Stanhope & Daida, 1998). In the DBMP, an algorithm needs to find solutions that minimize the Hamming distance to an arbitrary target pattern that may change over time. Given a solution $\vec{x} \in \{0,1\}^L$ (L is the length of the binary encoding of a solution for the problem) and the target pattern $\vec{a} \in \{0,1\}^L$ at time t, the Hamming distance between them is calculated as follows:

$$d_{\text{Hamming}}(\vec{x}(t), \vec{a}(t)) = \sum_{i=1}^{i=L} |x_i(t) - a_i(t)|. \tag{2.2}$$

The dynamics (i.e. dynamic characteristics) of the DBMP is determined by two parameters, g and d, which control the number of generations between changes and the degree (Hamming distance) by which the target pattern $\vec{a}$ is altered (for each change, d distinct and randomly chosen bits in $\vec{a}$ are inverted), respectively. For example, setting $(g, d) = (0, 0)$ results in a stationary function whereas setting $(g, d) = (20, 5)$ means that every 20 generations, the target pattern $\vec{a}$ changes by 5 bits randomly.

2.3.2 *Dynamic knapsack problems.* The knapsack problem is a classical NP-hard COP, where the solution space belongs to the binary space. Given a set of items, each of which has a weight and a profit, and a knapsack with a fixed capacity, the problem aims to select items to fill up the knapsack to maximize the total profit while satisfying the capacity constraint of the knapsack. The knapsack problem has frequently been used to test the performance of metaheuristics in stationary environments (Wilbaut *et al.*, 2008), and its dynamic version has also been used by researchers to test the performance of metaheuristics in dynamic environments (see Lewis *et al.*, 1998; Mori *et al.*, 1997; Ng & Wong, 1995). In the DKP, the system dynamics can be constructed by changing the weights of items, the profits of items, and/or the knapsack capacity over time according to some rules, respectively. So, the DKP can be described as follows:

$$\max \quad f(\vec{x}(t), t) = \sum_{i=1}^{n} p_i(t) \cdot x_i(t), \tag{2.3}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i(t) \cdot x_i(t) \leqslant C(t), \tag{2.4}$$

where n is number of items, $\vec{x}(t) \in \{0,1\}^n$ is a solution at time t, $x_i(t) \in \{0,1\}$ indicates whether item i is included in the subset or not, $p_i(t)$ is the profit of item i at time t, and $w_i(t)$ is the weight of item i at time t and $C(t)$ is the capacity of knapsack at time t. The weight and profit of each item may be bounded in the range of $[l_{\mathrm{w}}, u_{\mathrm{w}}]$ and $[l_{\mathrm{p}}, u_{\mathrm{p}}]$, respectively, and the capacity of knapsack may be bounded in the range of $[l_{\mathrm{c}}, u_{\mathrm{c}}]$.

Similarly, the static multi-dimensional knapsack problem (MKP) belongs to the class of NP-complete problems, which has a wide range of real-world applications, such as cargo loading, selecting projects to fund, budget management, etc. In the MKP, we have a number of resources (knapsacks), each of which has a capacity, and a set of items, each of which has a profit and consumes some amount of each resource. The aim is to select items to maximize the total profit while satisfying the capacity constraints for all resources. The DMKP has recently been used to investigate the performance of EAs for DCOPs (see Branke *et al.*, 2006; Uyar & Uyar, 2009). As in DKP, the profit and resource consumption for each item as well as the capacity of each resource may change over time in DMKP. Let $\vec{r}$, $\vec{p}$ and $\vec{c}$ denote the resource consumptions of items, the profits of items, and the capacities of resources, respectively, then the DMKP can be defined as follows:

$$\max \quad f(\vec{x}(t), t) = \sum_{i=1}^{n} p_i(t) \cdot x_i(t), \tag{2.5}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} r_{ij}(t) \cdot x_i(t) \leqslant c_i(t), \quad j = 1, 2, \ldots, m, \tag{2.6}$$

where n, $x_i(t)$, and $p_i(t)$ are as defined above, m is the number of resources, $r_{ij}(t)$ denotes the resource consumption of item i for resource j at time t and $c_i(t)$ is the capacity constraint of resource i at time t. The system dynamics can be constructed by changing the profits of items, resource consumptions of items and the capacity constraints of resources within certain upper and lower bounds over time according to some rules, respectively.

2.3.3 *The XOR DOP generator.* In Yang (2003) and Yang & Yao (2005), an XOR DOP generator that can generate dynamic environments from any binary encoded stationary problem using a bit-wise exclusive-or (XOR) operator has been proposed. Given a stationary problem $f(\vec{x})$ $(\vec{x} \in \{0,1\}^L$, where

L is the length of binary encoding, DOPs can be constructed from it as follows. Suppose the environment is changed every τ generations. For each environmental period k, an XORing mask $\vec{M}(k)$ is first incrementally generated as follows:

$$\vec{M}(k) = \vec{M}(k-1) \oplus \vec{T}(k), \tag{2.7}$$

where '$\oplus$' is the XOR operator (i.e. $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$) and $\vec{T}(k)$ is an intermediate binary template randomly created with $\rho \times L$ ($\rho \in [0.0, 1.0]$) ones inside it for environmental period k. Initially, $\vec{M}(0)$ is set to a zero vector. Then, the individuals at generation t are evaluated as follows:

$$f(\vec{x}(t), t) = f(\vec{x}(t) \oplus \vec{M}(k)), \tag{2.8}$$

where $k = \lfloor t/\tau \rfloor$ is the environmental period index.

With this XOR DOP generator, the environmental dynamics can be tuned by two parameters: τ controls the speed of environmental changes, while ρ controls the severity of changes. The bigger the value of τ, the slower the environment changes. The bigger the value of ρ, the more severe the environment changes.

The aforementioned XOR DOP generator in fact can construct *random dynamic environments* because there is no guarantee that the environment will return to a previous one after certain changes. The XOR DOP generator has been extended to construct *cyclic dynamic environments* in Yang (2005) and *cyclic dynamic environments with noise* further in Yang & Yao (2008).

This XOR DOP generator has two properties. One is that the distances among the solutions in the search space remains unaltered after an environmental change. The other is that the properties of the fitness landscape are not changed after an environmental change, which facilitates the analysis of the behaviour of algorithms.

2.3.4 *Dynamic travelling salesman problems.* The TSP is another classical NP-complete COP. It can be described as follows: Given a set of cities, we need to find the shortest route that starts from one city and visits each of the other cities once and only once before returning to the starting city.

DTSPs have a wide range of real-world applications, especially in the optimization of dynamic networks, like network planning and designing, load-balance routing and traffic management. There are several variations of DTSPs proposed in the literature, e.g. changing the topology of cities by replacing cities (see Guntsch & Middendorf, 2002; Mavrovouniotis & Yang, 2010, 2011b), and changing the distances between cities by adding traffic factors to the links between cities (see Mavrovouniotis & Yang, 2011a,d).

In Li *et al.* (2006), a DTSP is defined as a TSP with a dynamic cost (distance) matrix as follows:

$$D(t) = \{d_{ij}(t)\}_{n*n}, \tag{2.9}$$

where n is the number of cities and $d_{ij}(t)$ is the cost from city i to city j. The objective is to find a minimum-cost route containing all cities at time t. DTSP can be defined using a function $f(x, t)$ as follows:

$$f(x, t) = \text{Min}\left(\sum_{i=1}^{n} d_{x_i, x_{i+1}}(t)\right), \tag{2.10}$$

where $x_i \in \{1, 2, \ldots, n\}$ denotes the ith city in the solution such that $x_{n+1} = x_1$ and $\forall i, j:$ if $i \neq j$, $x_i \neq x_j$.

2.3.5 *Dynamic multicast routing problems in MANETs.* Cheng & Yang (2010b) studied the dynamic multicast routing problems in MANETs, where MANETs were modelled within a fixed geographical graph, which is composed by a set of wireless nodes and a set of communication links connecting two neighbour nodes that fall into the radio transmission range. To simulate the wireless network in dynamic environments in the real world, two different kind of models were created, which are the general dynamics model and the worst dynamics model, respectively. In the general dynamics model, periodically or randomly, some nodes are scheduled to sleep or some sleeping nodes are scheduled to wake up due to the energy conservation. While, in the worst model, each change is produced manually by removal of a few links on the current best multicast tree.

The models can be described by a MANET $G(V,E)$ and a multicast communication request from node s to a set of receivers R with a delay upper bound δ. So, the dynamic delay-constrained multicast routing problem is to find a series of trees $\{T_i|i \in \{0,1,\ldots\}\}$ over a series of graphs $\{G_i|i \in \{0,1,\ldots\}\}$, which satisfy the delay constraint and have the least tree cost as follows:

$$\max_{r_j \in R} \left\{ \sum_{l \in P_T(s,r_j)} d_l \right\} \leqslant \delta,$$

$$C(T_i) = \min_{T \in G_i} \left\{ \sum_{l \in T(V_T,E_T)} c_l \right\},$$

where $G_i(V_i,E_i)$ is the MANET topology after the ith change, $R = \{r_0, r_1, \ldots, r_m\}$ is a set of receivers of a multicast request, $T_i(V_{T_i}, E_{T_i})$ is a multicast tree with nodes V_{T_i} and links E_{T_i}, $P_T(s,r_j)$ is a path from s to r_j on the tree T_i, d_l represents the transmission delay on the communication link l, C_{T_i} is the cost of the tree T_i, and $\delta(P_i)$ is the total transmission delay on the path P_i.

2.3.6 *Dynamic shortest path routing problems in MANETs.* Yang *et al.* (2010) studied the DSPRP in MANETs, where MANETs were modelled as above within a fixed geographical graph that consists of wireless nodes and communication links. A communication link $l(i,j)$ cannot be used for packet transmission unless both node i and node j have a radio interface each with a common channel. In addition, message transmission on a wireless communication link will incur remarkable delay and cost.

The DSPRP can be informally described as follows. Initially, given a network of wireless routers, a delay upper bound, a source node and a destination node, we wish to find a delay-bounded least cost loop-free path on the topology graph. Since the end-to-end delay is a pretty important quality-of-service (QoS) metric to guarantee the real-time data delivery, the routing path is required to satisfy the delay constraint. Then, periodically or stochastically, due to energy conservation or some other issues, some nodes are scheduled to sleep or some sleeping nodes are scheduled to wake up. Therefore, the network topology changes from time to time. The objective of the DSPRP is to quickly find the new optimal delay-constrained least cost acyclic path after each topology change.

More formally, a MANET can be modelled by an undirected and connected topology graph $G(V,E)$, where V represents the set of wireless nodes (i.e. routers) and E represents the set of communication links connecting two neighbouring routers falling into the radio transmission range. A unicast communication request from the source node s to the destination node r with the delay upper bound $\varDelta$. Let $G_0(V_0,E_0)$ denote the initial MANET topology graph and $G_i(V_i,E_i)$ denote the MANET topology graph after the ith change. The *delay-constrained DSPRP* is to find a series of paths $\{P_i|i \in \{0,1,\ldots\}\}$ over a series of graphs $\{G_i(V_i, E_i) | i \in \{0, 1, \ldots\}\}$, which satisfy the delay constraint as shown in Equation (2.11) and have the least path cost as shown in Equation (2.12).

$$\Delta(P_i) = \sum_{l \in P_i(s,r)} d_l \leqslant \Delta, \tag{2.11}$$

$$C(P_i) = \min_{P \in G_i} \left\{ \sum_{l \in P_i(s,r)} c_l \right\}, \tag{2.12}$$

where $P_i(s, r)$ is a path from s to r on the graph G_i, c_l and d_l are the cost and transmission delay on the communication link l, respectively, $\Delta(P_i)$ is the total transmission delay on the path P_i, and $C(P_i)$ is the total cost of the path P_i.

2.4 *Characteristics of DCOPs*

As shown above, many benchmark and test DCOPs have been used in the literature. These DCOPs have different characteristics and can be classified into different groups based on the following criteria:

- Time-linkage: whether the future behaviour of the problem depends on the current solution found by an algorithm or not.
- Predictability: whether the changes are predictable or not.
- Visibility: whether the changes are visible to the optimization algorithm and, if so, whether changes can be detected by using just a few detectors.
- Constrained: whether the problem is constrained or not.
- Cyclicity: whether the changes are cyclic/recurrent in the search space or not.
- Factors that change: whether changes involve objective functions, domain of variables, number of variables, constraints and/or other parameters.

The common characteristics of the DCOPs in the literature are summarized as follows.

- Most DCOPs are non-time-linkage problems. There are only a couple of time-linkage test problems (see Bosman, 2007; Bosman & Poutré, 2007a).
- In the default settings of most DCOPs, changes are detectable by using just a few detectors. Due to their highly configurable property some benchmark generators can be configured to create scenarios where changes are more difficult to detect.
- In most cases, the factors that change are the objective functions. Exceptions are the problems by Cheng & Yang (2011, 2010a,b), Li & Yang (2008) and Mavrovouniotis & Yang (2010) where the dimension or problem size also changes.
- Many DCOPs have unpredictable changes in their default settings. Some of the generators/problems can be configured to allow predictable changes, at least in the frequency and periodicity of changes (see Simões & Costa, 2008).
- A majority of DCOPs have cyclic/recurrent changes.

- Most DCOPs assume periodical changes, i.e. changes occur every fixed number of generations or fitness evaluations. An exceptional work is that Simões & Costa (2008) studied the cases where the changes occur in some time pattern.

Further details on the charactersitcs of DCOPs, and DOPs, in general, can be found in Nguyen (2011).

3. Metaheuristics for DCOPs: performance measures and methodologies

For stationary COPs, usually, the goal of a metaheuristic algorithm is to find the global optimum as fast and precisely as possible, and the performance measures used to evaluate an algorithm are the convergence speed and the rate of achieving the optimum over multiple runs. However, for metaheuristics for DCOPs, the goal of an algorithm turns from finding the global optimum to tracking the changing optima as closely as possible over time. This requires an algorithm to firstly detect the changes and secondly track the changing optima (local optima or ideally the global optimum). In addition, in case the problems before and after a change somehow correlate with each other, an optimization algorithm is also expected to learn from its previous search experience as much as possible to hopefully progress the search more effectively. Otherwise, the optimization process after each change will simply become the process of solving a different problem starting from the old population.

The new requirements raised by DCOPs, i.e. tracking changing optima and transferring knowledge from previous search experience, imply that it would be better to use a population of individuals/searchers instead of a single individual (see Cruz *et al.*, 2011). Hence, over the last two decades, researchers have mainly applied population-based metaheuristics, e.g. GAs (see Cheng & Yang, 2010a,b), ACO (see Mavrovouniotis & Yang, 2010, 2011a,b,d), AISs (see Hart & Ross, 1999), estimation of distribution algorithms (see Fernandes *et al.*, 2008), to solve DCOPs (as well as other DOPs). Cruz *et al.* (2011) and Nguyen *et al.* (2012) have summarized different metaheuristics applied to general DOPs in the literature and relevant researchers.

The new requirements raised by DCOPs also make it difficult, if not infeasible, for the traditional metaheuristics to solve them satisfactorily. Hence, researchers need to develop new methodologies to modify metaheuristics to solve DCOPs and also develop/use new performance measures to evaluate the performance of metaheuristics for DCOPs. The following subsections will first briefly describe commonly used performance measures for evaluating metaheuristics for DCOPs and then review typical approaches that have been developed to enhance metaheuristics to satisfy the aforementioned requirements for solving DCOPs.

3.1 *Performance measures*

In order to compare metaheuristics for DOPs, researchers have also developed/used a number of different performance measures, e.g. accuracy by Trojanowski & Michalewicz (1999), collective mean fitness (CMF) by Morrison (2003), mean best-of-generation fitness by Yang & Yao (2003), offline error and offline performance by Branke (2001), Branke & Schmeck (2003), optimization accuracy, stability and reactivity by Weicker (2003). A brief summary of the most used measures for metaheuristics for DOPs is presented in Cruz *et al.* (2011, Table 4), where over 20 measures are listed. It is notable that many of these performance measures were proposed for evaluating generation-based metaheuristics, especially EAs, for DOPs. However, with proper modification, they are also applicable for evaluating non-generation-based metaheuristics for DOPs. Some of these measures are also suitable and applied for metaheuristics for DCOPs. The widely used performance measures for evaluating metaheuristics for

DCOPs are described as follows. A further detailed list of performance measures for DOPs and DCOPs can be found in Nguyen (2011, Section 2.2).

3.1.1 *Collective mean fitness.* This measure was first used by Morrison (2003) with the name of CMF and Yang & Yao (2003) with the name of mean best-of-generation fitness. This measure is calculated as the average for many runs of the best values at each generation on the same problem as follows:

$$\bar{F}_{\mathrm{BOG}} = \frac{1}{G} \times \sum_{i=1}^{i=G} \left(\frac{1}{R} \times \sum_{j=1}^{j=R} F_{\mathrm{BOG}_{ij}} \right), \tag{3.1}$$

where $\bar{F}_{\mathrm{BOG}}$ is the mean best-of-generation fitness, G is the total number of generations allowed for a run, R is the total number of runs and $F_{\mathrm{BOG}_{ij}}$ is the best-of-generation fitness of generation i of run j of an algorithm on a particular problem. The CMF measure is one of the most commonly used measures in metaheuristics for DCOPs.

The advantage of this measure is to enable algorithm designers to quantitatively compare the performance of algorithms. The disadvantage of the measure and its variants is that they are not normalized, hence can be biased by the difference of the fitness landscapes at different periods of change. For example, if at a certain period of change the overall fitness values of the landscape is particularly higher than those at other periods of changes, or if an algorithm is able to get particular high fitness value at a certain period of change, the final $\bar{F}_{\mathrm{BOG}}$ might be biased toward the high fitness values in this particular period and hence might not correctly reflect the overall performance of the algorithm. Similarly, if $\bar{F}_{\mathrm{BOG}}$ is used to evaluate the mean performance of algorithms in solving a group of problems, it is also biased toward problems with larger fitness values.

3.1.2 *Accuracy.* The *accuracy* measure was proposed by Trojanowski & Michalewicz (1999), which is calculated as the average of the smallest errors (the difference between the optimum value and the value of the best individual) achieved at the end of each change period (i.e. right before the moment of change), as follows:

$$E_{\mathrm{B}} = \frac{1}{m} \sum_{i=1}^{m} e_{\mathrm{B}}(i), \tag{3.2}$$

where m is the total number of changes for a run and $e_{\mathrm{B}}(i)$ is the best error just before the ith change occurs.

This measure is especially useful in situations where we are interested in the final solution that the algorithm achieved before the change. The measure also makes it possible to compare the final outcome of different algorithms. However, the measure also has three important disadvantages. First, it does not say anything about how the algorithms have done to achieve their performances. As a result, the measure is not suitable if what users are interested in is the overall performance or behaviours of the algorithms. Secondly, similar to the best-of-generation measure, this measure is also not normalized and hence can be biased towards periods where the errors are relatively large. Finally, the measure requires that the global optimum value at each change is known.

3.1.3 *Offline error and offline performance.* These two performance measures were proposed by Branke (2001) and Branke & Schmeck (2003). The *offline error* is measured as the average over, at each evaluation, the error of the best solution found since the last change of the environment. This measure is always greater than or equal to zero and would be zero for a perfect performance.

$$E^t_{\text{offline}} = \frac{1}{t}\sum_{j=1}^{t} e_{\text{best}}(j), \tag{3.3}$$

where t is the number of generations so far, and $e_{\text{best}}(j)$ is the best error gained by the algorithm at generation j since the last change.

Similar to the offline error measure, the *offline performance* measure is proposed to evaluate algorithms in case the exact values of the global optima are unknown, which is calculated as follows:

$$P^t_{\text{offline}} = \frac{1}{t}\sum_{j=1}^{t} f_{\text{best}}(j), \tag{3.4}$$

where t is the number of generations so far, and $f_{\text{best}}(j)$ is the best performance gained by the algorithm at generation j since the last change.

With this type of measures, the faster the algorithm to find a good solution, the higher the score. Similar to the $\bar{F}_{\text{BOG}}$, the offline error/performance are also useful in evaluating the overall performance of an algorithm and to compare the final outcomes of different algorithms. These measures, however, have some disadvantages. First, they require that the time when a change occurs is known. Secondly, similar to $\bar{F}_{\text{BOG}}$, these measures are also not normalized and hence can be biased under certain circumstances.

3.1.4 *Optimization accuracy.* The *optimization accuracy* measure (also known as the *relative error*) was initially proposed by Feng *et al.* (1997) and was adopted by Weicker (2002) for the dynamic case, which is calculated as follows:

$$\text{accuracy}^t = \frac{\text{Best}^t - \text{Min}^t}{\text{Max}^t - \text{Min}^t}, \tag{3.5}$$

where Best^t is the fitness of the best individual in the population at time t, $\text{Max}^t \in \mathbb{R}$ and $\text{Min}^t \in \mathbb{R}$ are the best and worst fitness values of the search space, respectively. The range of the accuracy measure is from 0 to 1, with a value of 1 and 0 represents the best and worst possible values, respectively.

The optimization accuracy have the same advantages as the $\bar{F}_{\text{BOG}}$ and E_{offline} in providing quantitative value and in evaluating the overall performance of algorithms. The measure has an advantage over $\bar{F}_{\text{BOG}}$ and E_{offline}: it is independent to fitness rescalings and hence has become less biased to those change periods where the difference in fitness becomes particularly large. The measure, however, has a disadvantage: it requires information about the absolute best and worst fitness values in the search space, which might not always be available in practical situations. In addition, as pointed by Weicker (2002) himself, the optimization accuracy measure is only well defined if the complete search space is not a plateau at any generation t, because otherwise the denominator of Equation (3.5) at t would be zero.

3.1.5 *Stability and robustness.* Some studies also developed measures to evaluate the ability of algorithms in restricting the drop of fitness when a change occurs, of which the most representative measures are *stability* by Weicker (2002) and *robustness* by Rand & Riolo (2005). The measure *stability* is evaluated by calculating the difference in the fitness-based *accuracy* measure (see Equation (3.5)) of the considered algorithm between each two time steps as follows:

$$\text{stability}^t = \max\{0, \text{accuracy}^{(t-1)} - \text{accuracy}^t\}, \tag{3.6}$$

where accuracy^t has already been defined in Equation (3.5).

The *robustness* measure, similar to the *stability* measure, also determines how much the fitness of the next generation of the EA can drop, given the current generation's fitness. The measure is calculated as the ratio of the fitness values of the best solutions (or the average fitness of the population) between each two consecutive generations.

3.1.6 *Reactivity, recover rate and absolute recover rate.*

Convergence speed after changes, or the ability of the algorithm to recover quickly after a change, is also an aspect that attracts the attention of various studies in metaheuristics for DOPs and DCOPs. Weicker (2002) also proposed a *reactivity* measure dedicated to evaluating the ability of an adaptive algorithm to react quickly to changes, which is defined as follows:

$$\text{reactivity}_\epsilon^t = \min\left\{ t' - t \,\middle|\, \frac{\text{accuracy}^{t'}}{\text{accuracy}^t} \geqslant (1-\epsilon) \right\} \cup \{\text{maxgen} - t\}, \tag{3.7}$$

where $t, t' \in \mathbb{N}$, $t < t' \leqslant \text{maxgen}$, ϵ is a preset precision parameter, and maxgen is the number of generations in population-based metaheuristics such as EAs. The *reactivity* measure has a disadvantage: it is only meaningful if there is actually a drop in performance when a change occurs; otherwise, the value of the measure is always zero and nothing can be said about how well the algorithm reacts to changes.

To provide more insights into the convergence behaviour of algorithms, recently a pair of measures, the *recovery rate* (RR) and the *absolute recovery rate* (ARR) were proposed by Nguyen & Yao (2011, 2012). The RR measure is used to analyse *how quick an algorithm can recover from an environmental change and to start converging on a new solution before the next change occurs*. The ARR measure is used to analyse *how quick an algorithm can start converging on the global optimum before the next change occurs*. The RR and ARR measures can be used together in an RR–ARR diagram (see Nguyen & Yao, 2012, Fig. 1) to indicate whether an algorithm is able to converge to the global optimum; or whether it suffers from slow or pre-mature convergence.

3.2 *Methodologies*

As mentioned before, DCOPs pose new challenges to the design of traditional metaheuristics. New approaches are needed to enhance traditional metaheuristics to solve DCOPs. A simplest way for a metaheuristic algorithm to deal with DCOPs is to restart it from scratch whenever a change is detected or informed. Although the restart scheme does work for some special cases (see Yang & Yao, 2005), for most cases, it is more efficient to develop other approaches that can make use of knowledge gathered from previously encountered environments. Below, we briefly review each class of approaches to enhancing metaheuristics for DCOPs. An example list of academic and real-world DCOPs that were solved using each approach is provided in Table 1.

TABLE 1 *Examples of academic and real-world DCOPs that were solved using dynamic optimization metaheuristics techniques*

Approach	Academic combinatorial problems	Real-world problems
Tracking	DKP (Randall, 2005; Moser & Hendtlass, 2006; Yang, 2008).	Take-off runway scheduling problem (Atkin *et al.*, 2008)
	Dynamic vehicle routing problem with time window (Housroum *et al.*, 2006)	Aircraft landing problem (Moser & Hendtlass, 2007)
	Dynamic OneMax problem (Yang, 2008)	Document stream modelling problem (Araujo & Merelo, 2007)
	Plateu function (Yang, 2008)	Dynamic route planning for car navigation in Japan (Kanoh, 2007; Kanoh & Hara, 2008)
	DTSP (Li *et al.*, 2006)	Evolvable hardware problem (Tawdross *et al.*, 2006)
	Various dynamic scheduling problems (Ouelhadj & Petrovic, 2009)	Adaptive contamination source identification problem (Liu, 2008)
Introducing/ maintaining diversity	Dynamic facility layout problem (Balakrishnan & Cheng, 2000)	Ship scheduling problem (Mertens *et al.*, 2006)
	DTSP (Eyckelhof *et al.*, 2002; Angus & Hendtlass, 2002; Mavrovouniotis & Yang, 2011d)	Hydrothermal scheduling problem for electricity generators (Deb *et al.*, 2007)
	DKP (via the XOR benchmark) (Yang & Yao, 2005; Yang, 2008)	Dimensioning and load balancing for multi-topology Interior Gateway Protocol traffic (Wang *et al.*, 2008)
	Dynamic decomposable unitation-based and deceptive Functions (Yang & Yao, 2008; Yang, 2005)	DSPRP in MANET (Cheng & Yang, 2010b)
	Dynamic OneMax (Yang, 2005, 2008)	
	NK problems (Yang, 2005, 2008)	
Memory	Oscillating 0–1 knapsack problem (Goldberg & Smith, 1987; Lewis *et al.*, 1998)	Document stream modelling problem (Araujo & Merelo, 2007)
	DKP (Simões & Costa, 2007; Yang, 2008, 2005)	Dynamic route planning for car navigation system in Japan (Kanoh, 2007; Kanoh & Hara, 2008)
	Dynamic OneMax (Simões & Costa, 2007; Yang, 2008, 2005)	
	Dynamic decomposable unitation-based functions (Yang & Yao, 2008)	
	Plateau function (Yang, 2008)	
	Royal Road problem (Yang, 2005, 2003)	
	Deceptive problems (Yang, 2005, 2003)	
Prediction	The inventory management problem (Bosman & Poutré, 2007a)	Dynamic vehicle routing/scheduling problems with real-time traffic information (Kim *et al.*, 2005; Eglese *et al.*, 2006; Kanoh, 2007; Kanoh & Hara, 2008)

Continued

TABLE 1 *Continued*

Approach	Academic combinatorial problems	Real-world problems
	Dynamic job-shop scheduling problem (Branke & Mattfeld, 2005)	Dynamic supply-chain configurations problem (Akanle & Zhang, 2008)
	Dynamic pickup problem (Bosman & Poutré, 2007b)	Airlift mission monitoring and dynamic rescheduling problem (Wilkins *et al.*, 2008)
	Dynamic bit matching problem (Simões & Costa, 2008)	Dynamically managing restaurant tables using constraints (Vidotto *et al.*, 2007)
	DKPs (Simões & Costa, 2009)	Dynamic assignment problem in P2P networks (Martinez *et al.*, 2008)
		The take-off runway scheduling (Atkin *et al.*, 2008)
		The dynamic dimensioning and load balancing for IGP network traffic (Wang *et al.*, 2008)
Multi-population	Dynamic oscillating 0/1 knapsack problem (Klinkmeijer *et al.*, 2006)	Adaptive QoS routing in MANET (Kotecha & Popat, 2007)
	DKP (Yang & Yao, 2003), Royal-road and deceptive functions (Yang & Yao, 2005)	DSPRP in MANET (Cheng & Yang, 2010b)
	Dynamic decomposable unitation-based functions (Yang & Yao, 2008)	
	Dynamic Royal-road, deceptive and scaling functions (Tinós & Yang, 2007)	

A more detailed survey of methodologies for solving DCOPs and DOPs can be found in Nguyen (2011, Section 2.1).

3.2.1 *Tracking.* The simplest, most obvious and commonly used technique in metaheuristics is tracking, i.e. to start from the previous optimal solutions found just before the change occurs. The purpose of tracking is two-fold. First, in case there is some correlation between the changes, starting from previous solutions may increase the chance to find the new optimum. Second, starting from existing solutions may minimize the risk of significantly changing the existing system, and hence reduce the operational cost. This purpose is particularly important in real-world applications (Nguyen, 2011).

In solving DCOPs using the tracking technique, the previously found solution is generally incrementally updated to adapt to the new change (Atkin *et al.*, 2008; Deb *et al.*, 2007; Kanoh, 2007; Kanoh & Hara, 2008; Moser & Hendtlass, 2007; Yang, 2008). If this newly updated solution becomes infeasible after a change, it is repaired by removing/replacing the components that make it infeasible (Moser & Hendtlass, 2006; Randall, 2005). The existing solution may need to be 'deconstructed' first to determine which of its components should be discarded due to constraint violation (Randall, 2005). The reusable parts of the existing solutions are then generally copied directly to all individuals or at least part of the population to deal with the new dynamic environment. In many cases, the fact that new solutions should be created from old solutions to minimize the amount of changing cost is even a requirement (Mertens *et al.*, 2006; Ngo *et al.*, 2006; Wang *et al.*, 2008). Furthermore, sometimes large deviations from previous solution are penalized (Atkin *et al.*, 2008; Beasley *et al.*, 2004). It was also suggested that tracking were prefered over restarting because it could help produce a good solution faster (Araujo &

Merelo, 2007). Experiments confirmed that tracking is able to provide better performance than restarting (Eyckelhof *et al.*, 2002; Mertens *et al.*, 2006) in the considered DCOPs.

3.2.2 *Introducing and/or maintaining diversity.* In stationary optimization, the convergence of a population-based metaheuristic is required so that an algorithm could focus on finding the best solution in the promising region. In dynamic optimization, however, convergence may result in negative effects, since if the dynamic landscape changes in one area and there is no member in the population for the algorithm in this area, the change will become undetected. As a result, it is impossible for a standard metaheuristic to detect a change once it has already converged.

Intuitively, one simple solution to this drawback is to increase/maintain the diversity of the population. In solving DCOPs, one of the common way to maintain diversity is to introduce random individuals (*random immigrants*) to the population (Balakrishnan & Cheng, 2000; Deb *et al.*, 2007; Cheng & Yang, 2010b; Yang & Yao, 2005) or to introduce the mutated versions of existing individuals (Deb *et al.*, 2007; Mavrovouniotis & Yang, 2011d; Yang, 2005, 2008). Another way is to bias the selection process so that less good but more diversified solutions are retained for the next generation. In solving the dynamic TSP using ant colony, Eyckelhof *et al.* (2002) reduced the pheromone levels of better roads and increased the pheromone levels in worse roads so that some of the worse roads will be chosen. The level of diversity can be adaptively controlled by the evolutionary process, as in Yang & Yao (2005, 2008).

3.2.3 *Memory.* If changes in DCOPs are cyclic or recurrent, i.e. the optima could return to the regions near to previous locations, then it would be useful to employ memory, i.e. to re-use previously found solutions to save computational time and bias the search process. The memory plays the role as a reserved place storing old solutions for maintaining diversity when needed. The memory can be integrated *implicitly* in EAs, or *explicitly* as a separate component. In the case of implicit memory, redundant coding using multiploid genomes (a chromosome contains two or more alleles (i.e., gene values) at each locus, each corresponds to a specific state of a dynamic environment) is the most commonly used approach for solving DCOPs, e.g. see Goldberg & Smith (1987), Lewis *et al.* (1998), Uyar & Harmanci (2005) and Yang (2006c). In these studies, whenever a change occurs the algorithm will choose among the available alleles the most suitable one for the current environmental state. Another interesting example of implicit memory is in Eyckelhof *et al.* (2002) (solving the DTSP) where existing solutions were prevented from being eliminated so that they could be used in the future. This was done by setting a lower boundary for the fitness values of existing solutions (previous found roads) so that they never approach zero.

In the case of using explicit memory to solve DCOPs, the memory can be maintained *directly* in the form of previous good solutions (Guntsch & Middendorf, 2002; Louis & Xu, 1996; Yang, 2005, 2006a; Yang & Yao, 2008; Yu & Suganthan, 2009; Mavrovouniotis & Yang, 2011d), or *indirectly* in the form of associative information. Various types of associative information can be included, e.g. the environment at a given time (Eggermont *et al.*, 2001); the list of environmental states and state transition probabilities (Simões & Costa, 2008); successful individuals for certain types of changes (Simões & Costa, 2003; Yang, 2006b); the probability vector that created the best solutions (Yang & Yao, 2008) or the distribution statistics of the population at a given time (Yang, 2006a).

Regarding how to update the memory, usually the best found elements (direct or associative) in a generation are used to update the memory. A newly found element replaces an existing element in the memory, which can be the oldest member in the memory (Eggermont *et al.*, 2001; Simões & Costa, 2007), the one with the least contribution to the diversity of the population (Eggermont *et al.*, 2001;

Simões & Costa, 2007; Yang, 2005; Yang & Yao, 2008), or the one with the least contribution to fitness (Eggermont *et al.*, 2001).

Once the memory has been established, during the search usually the best elements in the memory (i.e. the ones that show the best results when being re-evaluated) will be used to replace the worst individuals in the population.

Memory approaches are particularly useful for solving problems with cyclically changing environments. For example, Yang (2008) showed that the memory-based versions of GA and random-immigrant significantly outperform the original algorithms in cyclic DCOPs.

3.2.4 *Learning and prediction.* In cases where changes exhibit some predictable patterns, it might be sensible to try to learn these types of patterns from the previous search experience and based on these patterns try to predict changes in the future. Some studies have been produced following this idea to exploit the predictability of dynamic environments. On academic problems, most current prediction approaches have been applied to the continuous domain (see reviews in Nguyen *et al.*, 2012; Nguyen, 2011), but there are some research in the combinatorial domain as well. For example, on the dynamic bit matching and DKP, Simões & Costa (2008, 2009) predicted the future moment when the next change will occur and possible environments that will appear in the next change.

There are also some studies (Bosman, 2005, 2007; Bosman & Poutré, 2007b; Nguyen & Yao, 2009) on time-linkage problems, i.e. problems where the current solutions found by the algorithms can influence the future dynamics. In such problems, it was suggested that the only way to solve the problems effectively is to anticipate or predict future changes and take into account the possible future outcomes given the current solutions, when solving the problems online. Time-linkage prediction based on learning was made successfully on DCOPs such as the dynamic pickup problem and the inventory management problem (Bosman & Poutré, 2007a) and the dynamic supply-chain configuration problem (Akanle & Zhang, 2008). Anticipation of future changes based on expert knowledge was made on DCOPs such as the take-off runway scheduling problem (Atkin *et al.*, 2008), dynamic vehicle routing/scheduling problems with real-time traffic information (Eglese *et al.*, 2006; Kanoh, 2007; Kanoh & Hara, 2008; Kim *et al.*, 2005), dynamic dimensioning and load balancing problem for IGP network traffic (Wang *et al.*, 2008), airlift mission monitoring and dynamic rescheduling problem (Wilkins *et al.*, 2008), managing restaurant tables subject to constraints (Vidotto *et al.*, 2007) and dynamic assignment problem in P2P networks (Martinez *et al.*, 2008). Another related study is the anticipation approach (Branke & Mattfeld, 2005) in solving dynamic scheduling problems where in addition to finding good solutions, the solver also tries to move the system 'into a flexible state' where adaptation to changes can be done more easily.

For details of how the prediction and/or anticipation are implemented, readers are referred to Nguyen (2011, Chapter 3 and Tables 1–4 in the appendix).

3.2.5 *Multi-population.* Another approach, which to some extent can be seen as a combination of some or all of the previous approaches, uses multiple sub-populations concurrently in which each sub-population may handle a separate area of the search space and each of them may also take responsibility for a separate task.

One of the main goals of using multiple populations in solving DCOPs is to monitor multiple sub-optimal solutions in a hope that when a change occurs, one of such solution might become the global optimum, or at least becomes close to the global optimum. This also helps in maintaining diversity. In

Kotecha & Popat (2007) (dynamic QoS routing in MANETs), each sub-population focuses on monitoring the best routes to one different destination in the hope that when changes occurs this route might become (a part of) the global optimum. In Cheng & Yang (2010b); Klinkmeijer *et al.* (2006) (shortest path in MANETs and dynamic (oscillatory) 0/1 knapsack problem, respectively), each sub-population represents an area around a local optimum and when changes occur all the sub-populations will be reevaluated and updated.

Each sub-population might have a different role/responsibility. In Cheng & Yang (2010b) one sub-population is used for exploring, while others are used for exploiting/tracking. In Klinkmeijer *et al.* (2006), at one time there is only one active subpopulation while the rest function as memory and will only be reactivated when a change occurs. In Yang & Yao (2003) and Yang (2008), each subpopulation has a different search strategies, e.g. one has a higher diversity rate than another. In Tinós & Yang (2007), one subpopulation is dedicated to preserve diversified individuals generated for diversity purpose.

The multi-population approach is more commonly used in the continuous domain than in the combinatorial domain.

4. Case study

In previous sections, we have introduced/reviewed the domain of metaheuristics for DCOPs from different aspects and at a top level of viewpoint. In this section, in order to present a deeper and more comprehensive view on how to use a metaheuristic method to solve a DCOP, we present a case study taken from the literature on ACO algorithms with immigrants schemes for solving the DTSP with traffic factors by Mavrovouniotis & Yang (2011a,c,d).

4.1 *The DTSP with traffic factors*

Mavrovouniotis & Yang (2011a,c,d) investigated the DTSP with traffic factors, where the cost of the link between cities i and j is assumed to be $D_{ij} = D_{ij} \times F_{ij}$, where D_{ij} is the normal travelled distance and F_{ij} is the traffic factor between cities i and j. When the problem is to be changed (e.g. every f iterations of an ACO algorithm), a random number R in $[F_{\rm L}, F_{\rm U}]$ is generated probabilistically to represent traffic between cities, where $F_{\rm L}$ and $F_{\rm U}$ are the lower and upper bounds of the traffic factor, respectively. Each link has a probability m to add traffic such that $F_{ij} = 1 + R$, where the traffic factor of the remaining links is set to 1 (indicating no traffic). Note that f and m denote the frequency and magnitude of the changes in the dynamic environment, respectively. This type of environments are denoted *random DTSPs* because previously visited environments are not guaranteed to reappear.

Another variation of the DTSP with traffic factors is the DTSP where the dynamic changes occur with a cyclic pattern. This type of environments are denoted as *cyclic DTSPs* because previously visited environments will reappear in the future. A cyclic environment can be constructed by generating different dynamic cases with traffic factors as the base states, representing DTSP environments where each link has a probability m to add low, normal or high traffic as in random DTSPs. Then, the environment cycles among these base states, every f iterations, in a fixed logical ring order. Figure 1 illustrates the difference between a random DTSP and a cyclic DTSP.

4.2 *Standard ACO and population-based ACO algorithms for the DTSP*

The ACO algorithm was first proposed and applied for the stationary TSP (Colorni *et al.*, 1992; Dorigo, 1992), which imitates the behaviour of real ants when they search for food from their nest to a food source. Ants communicate using pheromone, which is a chemical substance produced by them and is

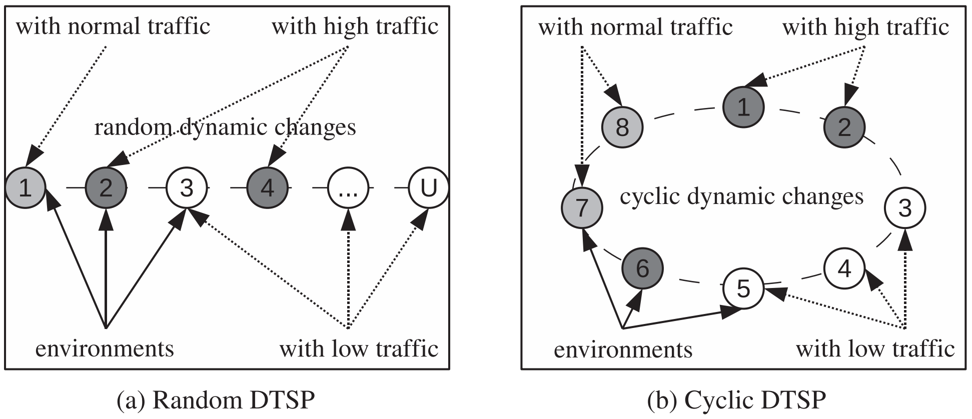

FIG. 1. Illustration of (a) a random DTSP with unlimited states in the dynamic environment and (b) a cyclic DTSP with eight states in the dynamic environment. Each node represents a state in the dynamic environment (or an environment in short), and white, light grey and dark grey represent low, medium and high traffic jams, respectively.

applied to their trails. The more pheromone on a specific trail, the higher the possibility of that trail to be followed by ants. Using this scheme, ants indirectly communicate and cooperate to complete their food searching task as efficiently as possible. Nowadays, ACO algorithms have been successfully applied for different COPs (Dorigo & Stützle, 2004).

The standard ACO (S-ACO) algorithm (i.e. Max–Min AS (MMAS)) (see Stuetzle & Hoos, 2000), consists of a population of μ ants. Initially, all ants are placed to a randomly selected city and all pheromone trails are initialized with an equal amount of pheromone. With a probability $1 - q_0$, where $0 \leqslant q_0 \leqslant 1$ is a parameter of the decision rule, ant k chooses the next city j, while its current city is i, probabilistically, as follows:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \quad \text{if } j \in N_i^k, \tag{4.1}$$

where τ_{ij} and $\eta_{ij} = 1/D_{ij}$ are the existing pheromone trails and heuristic information available *a priori* between cities i and j, respectively, N_i^k denotes the neighbourhood of cities of ant k that have not yet been visited when its current city is i, and α and β are the two parameters that determine the relative influence of pheromone trail and heuristic information, respectively. With the probability q_0, ant k chooses the next city with the maximum probability, i.e. $[\tau]^\alpha[\eta]^\beta$, and not probabilistically as in Equation (4.1).

Later on, the best ant retraces the solution and deposits pheromone according to its solution quality on the corresponding trails as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{\text{best}} \quad \forall (i,j) \in T^{\text{best}}, \tag{4.2}$$

where $\Delta\tau_{ij}^{\text{best}} = 1/C^{\text{best}}$ is the amount of pheromone that the best ant deposits and C^{best} is the tour cost of T^{best}. However, before adding any pheromone, a constant amount of pheromone is deduced from all trails due to the pheromone evaporation such that, $\tau_{ij} \leftarrow (1-\rho)\tau_{ij}$, $\forall (i,j)$, where $0 < \rho \leqslant 1$ is the rate of evaporation. The pheromone trail values are kept to the interval $[\tau_{\min}, \tau_{\max}]$, where $\tau_{\min}$ and $\tau_{\max}$ denote the minimum and maximum pheromone amount, respectively, and they are re-initialized to $\tau_{\max}$

every time the algorithm shows a stagnation behaviour, where all ants follow the same path, or when no improved tour has been found for several iterations).

The population-based ACO (P-ACO) algorithm by Guntsch & Middendorf (2002) is the memory-based version of ACO. The algorithm maintains a population of K ants (solutions), called population-list (memory) and denoted as k_{long}, which is used to update pheromone trails without any evaporation. The initial phase and the first iterations of the P-ACO algorithm work in the same way as with the S-ACO algorithm. The pheromone trails are initialized with an equal amount of pheromone and the population-list is empty. For the first K iterations, the iteration-best ant deposits a constant amount of pheromone, using Equation (4.2) with $\Delta\tau_{ij}^{\text{best}} = (\tau_{\max} - \tau_{\text{init}})/K$, where τ_{init} denotes the initial pheromone amount. This positive update procedure is performed whenever an ant enters into the population-list. At iteration $K + 1$, the ant that has entered into the population-list first, i.e. the oldest ant, needs to be removed in order to make room for the new one, and its pheromone trails are reduced by $\Delta\tau_{ij}^{\text{best}}$, which equals to the amount added when it entered the population-list before. The population-list k_{long} is a long-term memory since it may contain ants from previous environments that survive in more than one iteration. Therefore, when a change occurs, the ants stored in k_{long} are re-evaluated in order to be consistent with the new environment where different traffic factors F_{ij} are introduced. The pheromone trails are updated accordingly using the ants currently stored in k_{long}.

4.3 *ACO algorithms with immigrants schemes for the DTSP*

Both S-ACO and P-ACO algorithms face the convergence problem when applied to the DTSP. For S-ACO, from the initial iterations the population of ants will eventually converge to a solution, and, thus, high intensity of pheromone trails will be generated on a single path. The pheromone trails will influence ants forwarding them to the best path in hand even after a dynamic change. For P-ACO, identical ants may be stored in the population-list and dominate the search space with a high intensity of pheromone to a single trail. In order to handle the problem faced by S-ACO and P-ACO algorithms when addressing DTSPs, immigrants schemes have been integrated with ACO, see Mavrovouniotis & Yang (2010, 2011a,d), since they maintain a certain level of diversity during the execution and transfer knowledge from previous environments.

The framework of ACO algorithms with immigrants schemes is inspired by the P-ACO framework. Considering that P-ACO maintains a population of solutions, immigrant ants can be generated to replace ants in the current population. The main idea is to generate the pheromone information for every iteration considering information from the pheromone trails of the previous environment and extra information from the immigrant ants generated. Therefore, instead of using a long-term memory k_{long} as in P-ACO, a short-term memory, denoted k_{short}, of size K_{s} is used, where all ants stored from iteration $t - 1$ are replaced by the first K_{s} best ants of the current iteration t, instead of only replacing the oldest one as in P-ACO. Moreover, a predefined number of immigrant ants are generated and replace the worst ants in k_{short}, where the number of immigrant ants is usually small in comparison to K_{s}. When new ants are added to k_{short}, a positive update is made to their pheromone trails, and when ants are removed from k_{short}, a corresponding negative update is made to their pheromone trails as in P-ACO.

Different ACO variants have been developed with the integration of immigrants schemes with P-ACO. They differ from each other in the way immigrants are generated. Mavrovouniotis & Yang (2010) investigated the random immigrants ACO (RIACO), elitism-based immigrants ACO (EIACO) and hybrid immigrants ACO (HIACO), where immigrant ants in RIACO and EIACO are generated randomly and using the best ant from k_{short}, respectively, and in HIACO part of the immigrants are generated randomly and part are generated using the best ant from k_{short}. In the memory-based immigrants

ACO (MIACO) studied in Mavrovouniotis & Yang (2011a), immigrant ants are generated using the best ant from k_{long}. In EIACO and MIACO, the information obtained from the elitism- and memory-based immigrants to transfer knowledge is based on individual information (i.e. one ant). In Mavrovouniotis & Yang (2011d), an environmental information-based immigrants ACO (EIIACO) algorithm was proposed where immigrants ants are generated using environmental information (i.e. all the ants stored in k_{short}) to transfer knowledge from the previous environment to a new one.

4.4 *Experimental results*

Mavrovouniotis & Yang (2011c) experimentally studied the above ACO variants on a set of DTSPs with traffic factors with different dynamics settings, including both random and cyclic environments, and compared their performance based on the CMF measure. From the experimental results and analyses, the following conclusions can be drawn.

- Immigrants schemes improve the performance of ACO for DTSPs.
- RIACO is advantageous in fast and significantly changing environments.
- EIACO is advantageous in slowly and slightly changing environments.
- HIACO promotes the performance of EIACO in slowly changing environments, while it slightly degrades the performance of RIACO in fast changing environments.
- MIACO is advantageous in cyclic changing environments, where previous environments will re-appear.
- EIIACO promotes the performance of EIACO in environments with significant changes.
- Transferring too much knowledge from previous environments may degrade the performance.
- A high level of diversity does not always improve the performance of ACO in DTSPs.

In general, almost all ACO algorithms with immigrants schemes outperform ACO algorithms without them. Furthermore, different immigrants schemes are beneficial in different dynamic environments.

4.5 *Steps on metaheuristics for DCOPs*

The above case study, though focusing on ACO algorithms for the DTSP, demonstrates general steps needed to apply/design a given metaheuristic for a specific DCOP, which may be summarized as follows.

Step 1: Model the specific DCOP using a proper formulation and representation.

Step 2: Specialize the given metaheuristic properly so that it is suitable for solving the stationary version of the specific DCOP.

Step 3: Modify the specialized metaheuristic properly via integrating different approaches described in Section 3.2 to improve its performance for the specific DCOP.

Step 4: Run the obtained metaheuristic to solve the specific DCOP.

Note that, in Step 3, we may use the domain knowledge to help us analyse the characteristics of the specific DCOP and decide which approaches to be integrated into our specialized metaheuristic. In Step 4, we may carry out preliminary experiments to further tune the obtained metaheuristic for the specific DCOP.

5. Challenges

In this section, we discuss some key challenges relevant to the domain of metaheuristics for DCOPs. Some of these challenges are specific to metaheuristics for DCOPs while some others apply to metaheuristics for general DOPs.

5.1 *Detecting changes*

For a metaheuristic to efficiently solve a DCOP, usually it needs to first detect changes and then take proper actions when they occur. So, how to detect changes effectively is an important issue for a metaheuristic method to solve DCOPs effectively. However, in the literature on metaheuristics for DCOPs, most studies assume that changes are easy to detect by a few detectors via monitoring possible changes in their fitness values, or a change becomes visible to an algorithm directly whenever it occurs. This assumption does not always hold. For many DCOPs, changes may be very difficult or even impossible to detect, see (Li, 2011, Chapter 6) and (Richter, 2009). For example, in a DTSP with traffic factors, if a change only affects those links that are not present in the detector solutions, it cannot be detected by the detectors. And if a change only affects those links that are not present in all individuals of the current population, it cannot be detected by the current population, which is more possible when the current population becomes converged. The DKP with changing capacity for the knapsack is another example where changes may be difficult to detect. Here, a change can be detected by monitoring the fitness of an individual over two successive generations if the capacity of the knapsack is reduced. But, on the other hand, if the capacity is increased, the change will go undetected.

In order to address the issue of hard-to-detect changes, there are two options. One is to design more effective detection techniques, e.g. increasing the number of detectors and distributing them in a wider area of the search space. However, this may involve more computational cost in terms of fitness evaluations. Another option, which may be better, is to build effective approaches into metaheuristics that do not need to detect changes at all, e.g. using diversity maintaining approaches instead of approaches that increase diversity after a change is detected.

5.2 *Understanding the characteristics of DCOPs*

In order to understand whether a metaheuristic method can solve a DCOP well or not, we need to understand what characteristics contribute to the hardness/easiness of a DCOP for metaheuristics, which is a challenging task for the domain. To this purpose, we need to not only understand the characteristics of the base COP from which a DCOP is constructed, but also study the relationship between the dynamics (e.g. changes in parameters) and changes in the fitness landscape. There have been recent studies devoted to this issue, of which some are briefly described below.

Branke *et al.* (2005) analysed the changes of the fitness landscape due to changes of the underlying problem instance and proposed a number of measures that are helpful to understand what aspects of the fitness landscape change and what information can be carried over from one stage of the problem to the next. They applied these measures to several instances of a DMKP, which resulted with interesting observations. Branke *et al.* (2006) further analysed the role of representation on the fitness landscape based on the DMKP.

Rohlfshagen & Yao (2008, 2010) analysed the properties of a dynamic subset sum problem and investigated the correlation between the dynamic parameters of the problem and the resulting movement of the global optimum. Interestingly, the authors showed empirically that the degree to which the global optimum moves in response to the underlying dynamics is correlated only in specific cases.

Tinos & Yang (2010) further analysed the properties of the XOR DOP generator based on the dynamical system approach of the GA proposed by Vose (1999), and showed that a DOP generated by the generator can be described as a DOP with permutation, where the fitness landscape is changed according to a permutation matrix. Hence, the XOR DOP generator can be simplified by moving the initial population of a change cycle instead of rotating each individual prior to each fitness evaluation.

Richter (2010) analysed and quantified the properties of spatio-temporal fitness landscapes constructed from CML using topological and dynamical landscape measures such as modality, ruggedness, epistasis, dynamic severity, and two types of dynamic complexity measures, Lyapunov exponents and bred vector dimension. Experiments were also carried out to study the relationship between these landscape measures and the performance criteria of an EA.

Nguyen (2011) carried out a detailed survey of real-world DOPs and DCOPs, and from this identified some important gaps between real-world DOPs and academic research, including the current coverage of metaheuristic academic research, the types of problems that have not been covered by the community, the characteristics and problem information that we can use to solve DOPs more effectively, and the way that DOPs are solved in real-world scenarios.

The above studies are interesting in terms of trying to understand what (types of) characteristics make a DCOP easy or difficult to solve by metaheuristics. However, these studies are just a first step into the road. Much more effort is needed along this challenging line of research for the domain of metaheuristics for DCOPs.

5.3 *Analysing the behaviour of an algorithm*

Another challenging issue on metaheuristics for DCOPs is to analyse the behaviour of algorithms. The research in the domain so far has mainly been empirical. Due to the difficulty, theoretical analysis of metaheuristics for DCOPs has just appeared in recent years with only a few results. These studies mainly focus on the tracking performance of simple metaheuristics, e.g. the (1+1) EA and (1+λ) EA,[1] on simple DCOPs, e.g. the DBMP.

As a first theoretical study, Stanhope & Daida (1999) analysed a (1+1) EA on the DBMP. They presented the transition probabilities of the (1+1) EA and showed that even small perturbations in the fitness function could have a significantly negative impact on the performance of the (1+1) EA. Droste (2002) analysed the first hitting time (FHT)[2] of a (1+1) evolution strategy (ES) on the DBMP, where exactly one bit is changed with a given probability after each function evaluation. Jansen & Schellbach (2005) presented a rigorous performance analysis of the $(1+\lambda)$ EA on a tracking problem in a two-dimensional lattice and showed that the expected FHT strictly increases with the offspring population size (i.e. λ) whereas the expected number of generations to reach the target decreases with λ. Weicker (2005) used Markov models to analyse the tracking behaviour of $(1, \lambda)$-ESs with different mutation operators for a DCOP with a single moving optimum. Recently, Rohlfshagen *et al.* (2009) analysed how the magnitude and frequency of change may affect the tracking performance of a (1+1) EA on two specially designed pseudo-Boolean functions under the framework of the XOR DOP generator by Yang (2003).

[1] In a (1+λ) EA, only one solution is maintained in the population. In each iteration, the unique solution acts as the parent to generate λ offspring via mutation. If the fitness of the best offspring is not worse than the parent, the best offspring will replace the parent; otherwise, the parent will survive into the next generation. The simplest $(1+\lambda)$ EA is (1+1) EA where $\lambda = 1$.

[2] The FHT of an EA is the time that the EA finds the optimal solution for the first time, and the expected FHT of an EA is the average time that the EA requires to find the optimal solution, which implies the average computational time complexity of the EA.

The theoretical studies in the domain, although not many so far, can serve as good starting points. Researchers need to take the challenge to analyse the behaviour of more advanced metaheuristics for more complex DCOPs regarding more performance measures.

6. Conclusion and future research directions

Many real-world optimization problems are DCOPs, where the problem involves changes over time. Addressing DCOPs is an important task. Due to the intrinsic characteristics, metaheuristics are good choice of tools to address DCOPs. Over the past two decades, DCOPs have been investigated by the metaheuristics community, and in recent years, DCOPs have attracted a growing interest from the metaheuristics community. This paper provides a tutorial on the domain of metaheuristics for DCOPs. We have covered the key aspects of the domain, including the definition and features of DCOPs, benchmark and test DCOPs studied in the literature, performance measures used to compare different metaheuristics for DCOPs, methodologies developed to improve the performance of metaheuristics for DCOPs, case studies on real-world applications and challenges.

In general, there have been some promising results considering the use of metaheuristics for DCOPs in the literature. However, the domain is still very young and there are many research directions to be pursued in the future.

- Modelling real-world DCOPs: Most studies in the domain so far have used general-purpose benchmark DCOPs, e.g. the DBMP, the DKP and the XOR DOP generator, to test and evaluate metaheuristics. These benchmark DCOPs are mainly academic problems, designed for the good of testing some properties of algorithms. For this purpose, we still need more academic DCOPs that can be used to test more different properties of algorithms, e.g. the dynamic modular functions by Rohlfshagen & Yao (2010) to test the role of modularity on EAs for DCOPs. However, to make the domain more meaningful and useful, we need to study real-world DCOPs. As shown in this paper, there have been some application studies on metaheuristics for real-world DCOPs, see Cheng & Yang (2010a,b), Chitty & Hernandez (2004), Xing *et al.* (2011) and Nguyen (2011). However, the number of real-world application studies so far is significantly below expectation. Researchers in the domain need to consider and model more real-world DCOPs, which is a challenging task, and apply metaheuristics to solve them in the future.

- Developing new methods for the domain: As mentioned before, for the moment, studies mainly focus on population-based metaheuristics, e.g. EAs and ACO algorithms, for DCOPs. On the one hand, we may need to develop more efficient population-based metaheuristics for DCOPs. On the other hand, we may need to study non-population-based metaheuristics for DCOPs or integrate them into population-based metaheuristics for DCOPs. Moreover, although several approaches have been developed to enhance traditional metaheuristics for solving DCOPs, new efficient approaches are still greatly needed to address different types of DCOPs. Different approaches have different strengths and weaknesses for metaheuristics for different DCOPs. Hence, it is also worthy to develop and investigate hybrid approaches to metaheuristics for DCOPs in the future.

- Theoretical research: As mentioned before, the theoretical studies in the domain are quite limited so far, due to the difficulty behind. The relative lack of theoretical results makes it hard to fully justify the strengths and weaknesses of metaheuristics and predict their behaviour for different DCOPs. Hence, theoretical studies are greatly needed for the domain. As reviewed, the analysis on the characteristics of DCOPs has recently started with a few results. But, this line of research needs to be enhanced significantly in order to gain insights as to what DCOPs are hard or easy for what types of metaheuristics. The analysis of dynamic behaviour of metaheuristics for DCOPs also needs to be pursued or enhanced. For many real-world DCOPs, it is more useful to know how well an algorithm tracks the moving optima within certain acceptable error levels rather than whether and how fast the algorithm could hit the moving optima. Hence, it is also important to analyse metaheuristics for DCOPs regarding such properties as tracking error and robustness.

Acknowledgements

The authors are grateful to the anonymous Associate Editor and reviewers for their thoughtful suggestions and constructive comments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E058884/1, Grant EP/E060722/2, and EP/K001310/1.

Funding

References

Akanle, O. & Zhang, D. (2008) Agent-based model for optimising supply-chain configurations. *Int. J. Prod. Econ.*, **115**, 444–460.

Angus, D. & Hendtlass, T. (2002) Ant colony optimisation applied to a dynamically changing problem. *Proceedings of the 15th International Conference on Industrial and Engineering, Applications of Artificial Intelligence and Expert Systems*. Springer-Verlag: London, pp. 618–627.

Aragon, V. S. & Esquivel, S. C. (2004) An evolutionary algorithm to track changes of optimum value locations in dynamic environments. *J. Comput. Sci. and Tech.*, **4**, 127–134.

Araujo, L. & Merelo, J. J. (2007) A genetic algorithm for dynamic modelling and prediction of activity in document streams. *Proceedings of the 2007 Genetic and Evolutionary Computation Conference, GECCO'07*, New York, NY: ACM, pp. 1896–1903.

Atkin, J. A. D., Burke, E. K., Greenwood, J. S. & Reeson, D. (2008) On-line decision support for take-off runway scheduling with uncertain taxi times at london heathrow airport. *J. Sched.*, **11**, 323–346.

Bäck, T. (1998) On the behavior of evolutionary algorithms in dynamic environments. *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, Piscataway, NJ: IEEE Press, pp. 446–451.

Balakrishnan, J. & Cheng, C. H. (2000) Genetic search and the dynamic layout problem. *Comput. OR*, **27**, 587–593.

Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M. & Abramson, D. (2004) Displacement problem and dynamically scheduling aircraft landings. *J. Oper. Res. Soc.*, **55**, 54–65.

Bosman, P. A. N. (2005) Learning, anticipation and time-deception in evolutionary online dynamic optimization. *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization* (S. Yang & J. Branke eds), New York, NJ: ACM.

Bosman, P. A. N. (2007) Learning and anticipation in online dynamic optimization. *Evolutionary Computation in Dynamic and Uncertain Environments* (S. Yang, Y.-S. Ong & Y. Jin eds). Studies in Computational Intelligence, vol. 51. Berlin: Springer, pp. 129–152.

Bosman, P. A. N. & Poutré, H. L. (2007a) Inventory management and the impact of anticipation in evolutionary stochastic online dynamic optimization. *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, Piscataway, NJ: IEEE Press, pp. 268–275.

Bosman, P. A. N. & Poutré, H. L. (2007b) Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. *Proceedings of the 2007 Genetic and Evolutionary Computing Conference, GECCO'07*, New York, NY: ACM, pp. 1165–1172.

Branke, J. (2001) *Evolutionary Optimization in Dynamic Environments*. Norwell, MA: Kluwer.

Branke, J. & Mattfeld, D. (2005) Anticipation and flexibility in dynamic scheduling. *Int. J. Prod. Res.*, **43**, 3103–3129.

Branke, J., Orbayi, M. & Uyar, S. (2006) The role of representations in dynamic knapsack problems. *EvoWorkshops 2006: Application of Evolutionary Computation*. Lecture Notes in Computer Science, vol. 3907, Berlin: Springer, pp. 764–775.

Branke, J., Salihoglu, E. & Uyar, S. (2005) Towards an analysis of dynamic environments. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO'05*, New York, NY: ACM, pp. 1433–1439.

Branke, J. & Schmeck, H. (2003) Designing evolutionary algorithms for dynamic optimization problems. *Theory and Application of Evolutionary Computation: Recent Trends* (S. Tsutsui & A. Ghosh eds). Berlin: Springer, pp. 239–262.

Chazottes, J. & Fernandez, B. (2005) *Dynamics of Coupled Map Lattices and of Related Spatially Extended Systems*. Heidelberg: Springer.

Cheng, H. & Yang, S. (2010a) Genetic algorithms with immigrants schemes for dynamic multicast problems in mobile ad hoc networks. *Eng. Appl. Artif. Intell.*, **23**, 806–819.

Cheng, H. & Yang, S. (2010b) Multi-population genetic algorithms with immigrants scheme for dynamic shortest path routing problems in mobile ad hoc networks. *EvoApplications 2010: Application of Evolutionary Computation*, Lecture Notes in Computer Science, vol. 6024, Berlin: Springer, pp. 562–571.

Cheng, H. & Yang, S. (2011) Genetic algorithms with elitism-based immigrants for dynamic load balanced clustering problems in mobile ad hoc networks. *Proceedings of the 2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments*, Piscataway, NJ: IEEE Press, pp. 1–7.

Chitty, D. & Hernandez, M. (2004) A hybrid ant colony optimization technique for dynamic vehicle routing. *Proceedings of the 2004 International Conference on Genetic and Evolutionary Computation, GECCO'04*, pp. 48–59.

Colorni, A., Dorigo, M., & Maniezzo, V. (1992) Distributed optimization by ant colonies. *Proceedings of the European Conference on Artificial Life*, Cambridge, MA: The MIT Press, pp. 134–142.

Cremers, M. L. A. G., Haneveld, W. K. K. & van der Vlerk, M. H. (2010) A dynamic day-ahead paratransit planning problem. *IMA J. Manag Math.*, **21**, 195–211.

Cruz, C., Gonzalez, J. & Pelta, D. (2011) Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Comput.*, **15**, 1427–1448.

Deb, K., Rao, U. B. & Karthik, S. (2007) Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*. Lecture Notes in Computer Science, vol. 4403, Berlin: Springer, pp. 803–817.

Dorigo, M. (1992) Optimization, learning and natural algorithms. *Ph.D. Thesis*, Politecnico di Milano, Italy.

Dorigo, M. & Stützle, T. (2004) *Ant Colony Optimization*. London, England: The MIT Press.

Droste, S. (2002) Analysis of the $(1+1)$ ea for a dynamically changing onemax-variant. *Proceedings of the 2002 IEEE Congress on Evolutionary Computation, CEC'02*, Piscataway, NJ: IEEE Press, pp. 55–60.

Eggermont, J., Lenaerts, T., Poyhonen, S. & Termier, A. (2001) Raising the dead; extending evolutionary algorithms with a case-based memory. *Proceedings of the 2001 International Conference on Genetic Programming*. Lecture Notes in Computer Science, vol. 2038, Berlin: Springer, pp. 280–290.

Eglese, R., Maden, W. & Slater, A. (2006) A road timetable TM to aid vehicle routing and scheduling. *Comput. Oper. Res.*, **33**, 3508–3519.

Eyckelhof, C. J., Snoek, M. & Vof, M. (2002) Ant systems for a dynamic tsp: Ants caught in a traffic jam. *Proceedings of the 3rd International Workshop on Ant Algorithms, ANTS 2002*. Lecture Notes in Computer Science, vol. 2463, Berlin: Springer, pp. 88–99.

Farmer, J., Packard, N. & Perelson, A. (1986) The immune system, adaptation and machine learning. *Physica D*, **22**, 187–204.

Feng, W., Brune, T., Chan, L., Chowdhury, M., Kuek, C. & Li, Y. (1997) Benchmarks for testing evolutionary algorithms. *Technical Report*. Center for System and Control, University of Glasgow.

Fernandes, C., Lima, C. & A.C., R. (2008) Umdas for dynamic optimization problems. *Proceedings of the 2008 Genetic and Evolutionary Computation Conference, GECCO'08*, New York, NY: ACM, pp. 399–406.

Garey, M. & Johnson, D. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Fransico: W. H. Freeman and Company.

Glover, F. (1986) Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, **13**, 533–549.

Goldberg, D. E. & Smith, R. E. (1987) Nonstationary function optimization using genetic algorithms with dominance and diploidy. *Proceedings of the International Conference on Genetic Algorithms*, Hillsdale, NJ: L. Erlbaum Associates Inc., pp. 59–68.

Guntsch, M. & Middendorf, M. (2002) Applying population-based ago to dynamic optimization problems. *Proceedings of the 3rd International Workshop on Ant Algorithms, ANTS 2002*. Lecture Notes in Computer Science, vol. 2463, Berlin: Springer, pp. 111–122.

Handa, H., Chapman, L. & Yao, X. (2007) Robust salting route optimization using evolutionary algorithms. *Evolutionary Computation in Dynamic and Uncertain Environments* (S. Yang, Y.-S. Ong, and Y. Jin eds). Berlin: Springer, pp. 497–517.

Hart, E. & Ross, P. (1999) An immune system approach to scheduling in changing environments. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference, GECCO'99*, pp. 1559–1566.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

Housroum, H., Hsu, T., Dupas, R. & Goncalves, G. (2006) A hybrid ga approach for solving the dynamic vehicle routing problem with time windows. *Proceedings of the 2nd International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA'06*, vol. 1, Piscataway, NJ: IEEE Press, pp. 787–792.

Jansen, T. & Schellbach, U. (2005) Theoretical analysis of a mutation-based evolutionary algorithm for a tracking problem in lattice. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO'05*, New York, NY: ACM, pp. 841–848.

Jin, Y. & Branke, J. (2005) Evolutionary optimization in uncertain environments—a survey. *IEEE Trans. Evol. Comput.*, **9**, 303–317.

Kanoh, H. (2007) Dynamic route planning for car navigation systems using virus genetic algorithms. *Int. J. Knowledge-Based Intell. Eng. Syst.*, **11**, 65–78.

Kanoh, H. & Hara, K. (2008) Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network. *Proceedings of the 2008 Genetic and Evolutionary Computation Conference, GECCO'08*, New York, NY: ACM, pp. 657–664.

Kim, S., Lewis, M. E. & III, C. C. W. (2005) State space reduction for non-stationary stochastic shortest path problems with real-time traffic information. *IEEE Trans. Intell. Trans. Syst.*, **6**, 273–284.

Kirkpatrick, S., Gelatt Jr., C. & Vecchi, M. P. (1983) Optimization by simulated annealing. *Science*, **220**, 671–680.

Klinkmeijer, L. Z., de Jong, E. & Wiering, M. (2006) A serial population genetic algorithm for dynamic optimization problems. *Proceedings of the 15th Belgian-Dutch Conference on Machine Learning, Benelearn'06*, pp. 41–48.

Kotecha, K. & Popat, S. (2007) Multi objective genetic algorithm based adaptive QoS routing in MANET. *Proceedings of the 2007 IEEE Congress on Evolutionary Computation, CEC'07*, pp. 1423–1428.

Lewis, J., Hart, E. & Ritchie, G. (1998) A comparison of dominance mechanisms and simple mutation on non-stationary problems. *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, PPSN V*, Berlin: Springer, pp. 139–148.

Li, C. (2011) Particle swarm optimisation in stationary and dynamic environments. *Ph.D. Thesis*, University of Leicester, UK.

Li, C. & Yang, S. (2008) A generalized approach to construct benchmark problems for dynamic optimization. *Proceedings of the 7th International Conference on Simulated Evolution and Learning, SEAl'08*. Lecture Notes in Computer Science, vol. 5361, Berlin: Springer, pp. 391–400.

Li, C., Yang, M. & Kang, L. (2006) A new approach to solving dynamic traveling salesman problems. *Proceedings of the 6th International Conference on Simulated Evolution and Learning, SEAL'06*, Berlin: Springer, pp. 236–243.

Li, C., Yang, S., Nguyen, T. T., Yu, E. L., Yao, X., Jin, Y., Beyer, H.-G. & Suganthan, P. N. (2008) Benchmark generator for cec 2009 competition on dynamic optimization. *Technical Report*. University of Leicester and University of Birmingham, UK.

Liekens, A. M. L. (2005) Evolution of finite populations in dynamic environments. *Ph.D. Thesis*, Technische Universität Eindhoven.

Liu, L. (2008) Real-time contaminant source characterization in water distribution systems. *Ph.D. Thesis*, North Carolina State University.

Louis, S. J. & Xu, Z. (1996) Genetic algorithms for open shop scheduling and re-scheduling. *Proceedings of the 11th International Conference on Computers and their Application*, Winona, MN: ISCA, pp. 99–102.

Martinez, M., Moron, A., Robledo, F., Rodriguez-Bocca, P., Cancela, H. & Rubino, G. (2008) A grasp algorithm using rnn for solving dynamics in a p2p live video streaming network. *Proceedings of the 8th International Conference on Hybrid Intelligent Systems, HIS'08*, Piscataway, NJ: IEEE Press, pp. 447–452.

Mavrovouniotis, M. & Yang, S. (2010) Ant colony optimization with immigrants schemes for dynamic environments. *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature, PPSN XI*. Lecture Notes in Computer Science, vol. 6239, Berlin: Springer, pp. 371–380.

Mavrovouniotis, M. & Yang, S. (2011a) Ant colony optimization algorithms with immigrants schemes for the dynamic travelling salesman problem. *Evolutionary Computation for Dynamic Optimization Problems* (S. Yang & X. Yao eds; to appear).

Mavrovouniotis, M. & Yang, S. (2011b) An immigrants scheme based on environmental information for ant colony optimization for the dynamic travelling salesman problem. *Proceedings of the 10th International Conference on Artificial Evolution*. Lecture Notes in Computer Science (to appear).

Mavrovouniotis, M. & Yang, S. (2011c) A memetic ant colony optimization algorithm for the dynamic traveling salesman problem. *Soft Comput.*, **15**, 1405–1425.

Mavrovouniotis, M. & Yang, S. (2011d) Memory-based immigrants for ant colony optimization in changing environments. *EvoApplications 2011: Application of Evolutionary Computation*, Berlin: Springer, pp. 1–10.

Mei, Y., Tang, K. & Yao, X. (2010) Capacitated arc routing problem in uncertain environments. *Proceedings of the 2010 IEEE Congress on Evolutionary Computation, CEC'10*, Piscataway, NJ: IEEE Press, pp. 1400–1407.

Mertens, K., Holvoet, T. & Berbers, Y. (2006) The DynCOAA algorithm for dynamic constraint optimization problems. *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY: ACM, pp. 1421–1423.

Mori, N., Imanishi, S., Kita, H. & Nishikawa, Y. (1997) Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. *Proceedings of the 1997 International Conference on Genetic Algorithms*, San Fransisco, CA: Morgan Kaufmann Publishers, pp. 299–306.

Morrison, R. (2003) Performance measurement in dynamic environments. *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems* (J. Branke ed.)., pp. 5–8.

Morrison, R. W. (2004) *Designing Evolutionary Algorithms for Dynamic Environments*. Berlin: Springer.

Moser, I. & Hendtlass, T. (2006) Solving problems with hidden dynamics: Comparison of extremal optimisation and ant colony system. *Proceedings of the 2006 IEEE Congress on Evolutionary Computation, CEC'06*, Piscataway, NJ: IEEE Press, pp. 1248–1255.

Moser, I. & Hendtlass, T. (2007) Solving dynamic single-runway aircraft landing problems with extremal optimisation. *Proceedings of the 2007 IEEE Symposium Series on Computational Intelligence in Scheduling*, Piscataway, NJ: IEEE Press, pp. 206–211.

Ng, K. P. & Wong, K. C. (1995) A new diploid scheme and dominance change mechanism for non-stationary function optimization. *Proceedings of the 6h International Conference on Genetic Algorithms*, San Fransisco, CA: Morgan Kaufmann Publishers, pp. 159–166.

Ngo, S. H., Jiang, X., Le, V. T. & Horiguchi, S. (2006) Ant-based survivable routing in dynamic wdm networks with shared backup paths. *J. Supercomputing*, **36**, 297–307.

Nguyen, T. T. (2011) Continuous Dynamic Optimisation Using Evolutionary Algorithms. *Ph.D. Thesis*, School of Computer Science, University of Birmingham. http://etheses.bham.ac.uk/1296 and http://www.staff.ljmu.ac.uk/enrtngu1/theses/phd_thesis_nguyen.pdf.

Nguyen, T. T., Yang, S. & Branke, J. (2012) Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*. *Published online first: 22 May 2012 (DOI: 10.1016/j.swevo. 2012.05.001)*.

Nguyen, T. T. & Yao, X. (2009) Dynamic time-linkage problem revisited. *EvoWorkshops 2009: Application of Evolutionary Computation*. Lecture Notes in Computer Science, vol. 5484, Berlin: Springer, pp. 735–744.

Nguyen, T. T. & Yao, X. (2011) Solving dynamic constrained optimisation problems using repair methods. *IEEE Trans. on Evol. Comput.*, submitted. http://www.staff.ljmu.ac.uk/enrtngu1/Papers/Nguyen_Yao_dRepair GA.pdf.

Nguyen, T. T. & Yao, X. (2012) Continuous dynamic constrained optimisation—the challenges. *IEEE Trans. on Evol. Comput.*, Accepted, [online]. http://www.staff.ljmu.ac.uk/enrtngu1/Papers/Nguyen_Yao_DCOP.pdf.

Ouelhadj, D. & Petrovic, S. (2009) Survey of dynamic scheduling in manufacturing systems. *J. Scheduling*, **12**, 417–431.

Rand, W. & Riolo, R. (2005) Measurements for understanding the behavior of the genetic algorithm in dynamic environments: a case study using the shaky ladder hyperplane-defined functions. *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization* (S. Yang & J. Branke eds).

Randall, M. (2005) A dynamic optimisation approach for ant colony optimisation using the multidimensional knapsack problem. *Recent Advances in Artificial Life*. Advances in Natural Computation, vol. 3. New Jersey: World Scientific, pp. 215–226.

Richter, H. (2004) Behavior of evolutionary algorithms in chaotically changing fitness landscapes. *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, PPSN VIII*, Lecture Notes in Computer Science, vol. 3242, Berlin: Springer, pp. 111–120.

Richter, H. (2009) Detecting change in dynamic fitness landscapes. *Proceedings of the IEEE 2009 Congress on Evolutionary Computation, CEC2009*, Piscataway, NJ: IEEE Press, pp. 1613–1620.

Richter, H. (2010) Evolutionary optimization and dynamic fitness landscapes: from reaction-diffusion systems to chaotic cml. *Evolutionary Algorithms and Chaotic Systems*. Berlin: Springer, pp. 409–446.

Rohlfshagen, P., Lehre, P. K. & Yao, X. (2009) Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change. *Proceedings of the 2009 Genetic and Evolutionary Computation Conference, GECCO'09*, pp. 1713–1720.

Rohlfshagen, P. & Yao, X. (2008) Attributes of dynamic combinatorial optimisation. *Proceedings of the 10th International Conference on Parallel Problem Solving From Nature, PPSN'08*. Lecture Notes in Computer Science, vol. 5361, Berlin: Springer, pp. 442–451.

Rohlfshagen, P. & Yao, X. (2010) On the role of modularity in evolutionary dynamic optimisation. *Proceedings of the 2010 IEEE World Congress On Computational Intelligence, WCCI 2010*, Piscataway, NJ: IEEE Press, pp. 3539–3546.

Rohlfshagen, P. & Yao, X. (2011) Dynamic combinatorial optimisation problems: an analysis of the subset sum problem. *Soft Comput.*, **15**, 1723–1734.

Simões, A. & Costa, E. (2003) An immune system-based genetic algorithm to deal with dynamic environments: Diversity and memory. *Proceedings of the 6th International Conference on Neural Networks and Genetic Algorithms, ICANNGA03*, Berlin: Springer, pp. 168–174.

Simões, A. & Costa, E. (2007) Improving memory's usage in evolutionary algorithms for changing environments. *Proceedings of the 2007 IEEE Congress on Evolutionary Computation, CEC'07*, Piscataway, NJ: IEEE Press, pp. 276–283.

Simões, A. & Costa, E. (2008) Evolutionary algorithms for dynamic environments: Prediction using linear regression and markov chains. *Proceedings of the 10th Int. Conf. on Parallel Problem Solving from Nature, PPSN X*, Lecture Notes in Computer Science 5199, pp. 306–315.

Simões, A. & Costa, E. (2009) Improving prediction in evolutionary algorithms for dynamic environments. *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, GECCO'09*, New York, NY: ACM, pp. 875–882.

STANHOPE, S. A. & DAIDA, J. M. (1998) Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment. *Proceedings of the 7th International Conference on Evolutionary Programming*. Lecture Notes in Computer Science, vol. 1447, Berlin: Springer, pp. 693–702.

STANHOPE, S. A. & DAIDA, J. M. (1999) $(1+1)$ genetic algorithm fitness dynamics in a changing environments. *Proceedings of the 1999 IEEE Congress on Evolutionary Computation, CEC'99*, vol. 3, Piscataway, NJ: IEEE Press, pp. 1851–1858.

STUETZLE, T. & HOOS, H. (2000) MAX-MIN ant system. *Future Generation Comput. Syst.*, **8**, 889–914.

TAWDROSS, P., LAKSHMANAN, S. K. & KONIG, A. (2006) Intrinsic evolution of predictable behavior evolvable hardware in dynamic environment. *Proceedings of the 6th International Conference on Hybrid Intelligent Systems, HIS'06*, Piscataway, NJ: IEEE Press, pp. 60.

TINÓS, R. & YANG, S. (2007) A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genet. Program. Evol. Mach.*, **8**, 255–286.

TINOS, R. & YANG, S. (2010) An analysis of the XOR dynamic problem generator based on the dynamical system. *Proceedings of the 11th International Conference on Parallel Problems Solving from Nature, PPSN XI*. Lecture Notes in Computer Science, vol. 6238, Part I, Berlin: Springer, pp. 274–283.

TROJANOWSKI, K. & MICHALEWICZ, Z. (1999) Searching for optima in non-stationary environments. *Proceedings of the 1999 IEEE Congress on Evolutionary Computation, CEC'99*, vol. 3, Piscataway, NJ: IEEE Press, pp. 1843–1850.

UYAR, A. S. & HARMANCI, A. E. (2005) A new population based adaptive domination change mechanism for diploid genetic algorithms in dynamic environments. *Soft Comput.*, **9**, 803–814.

UYAR, S. & UYAR, H. (2009) A critical look at dynamic multi-dimensional knapsack problem generation. *EvoWorkshops 2009: Application of Evolutionary Computation*. Lecture Notes in Computer Science, vol. 5484, Berlin: Springer, pp. 762–767.

VIDOTTO, A., BROWN, K. N. & BECK, J. C. (2007) Managing restaurant tables using constraints. *Knowledge-Based Syst*, **20**, 160 – 169.

VOSE, M. (1999) *The Simple Genetic Algorithm: Foundations and Theory*. Cambridge, MA: The MIT Press.

WANG, N., HO, K.-H. & PAVLOU, G. (2008) Adaptive multi-topology igp based traffic engineering with near-optimal network performance. *Proceedings of the 2008 IFIP Networking Conference*. Lecture Notes in Computer Science, vol. 4982, Berlin: Springer, pp. 654–666.

WEICKER, K. (2000) An analysis of dynamic severity and population size. *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, PPSN VI*. Lecture Notes in Computer Science, vol. 1917, Berlin: Springer, pp. 159–168.

WEICKER, K. (2002) Performance measures for dynamic environments. *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, PPSN VII*. Lecture Notes in Computer Science, vol. 2439, Berlin: Springer, pp. 64–73.

WEICKER, K. (2003) *Evolutionary Algorithms and Dynamic Optimization Problems*. Berlin: Der Andere.

WEICKER, K. (2005) Analysis of local operators applied to discrete tracking problems. *Soft Comput.*, **9**, 778–792.

WEISE, T., PODLICH, A., REINHARD, K., GORLDT, C. & GEIHS, K. (2009) Evolutionary freight transportation planning. *EvoWorkshops 2009: Application of Evolutionary Computation*. Lecture Notes in Computer Science 5484, Berlin: Springer, pp. 768–777.

WILBAUT, C., HANAFI, S., & SALHI, S. (2008) A survey of effective heuristics and their application to a variety of knapsack problems. *IMA J. Manag. Math.*, **19**, 227–244.

WILKINS, D. E., SMITH, S. F., KRAMER, L. A., LEE, T. J. & RAUENBUSCH, T. W. (2008) Airlift mission monitoring and dynamic rescheduling. *Eng. Appl. Artif. Intell.*, **21**, 141–155.

WOLDESENBET, Y. G. & YEN, G. G. (2009) Dynamic evolutionary algorithm with variable relocation. *IEEE Trans. Evol. Comput.*, **13**, 500–513.

XING, L., ROHLFSHAGEN, P., CHEN, Y. & YAO, X. (2011) A hybrid ant colony optimisation algorithm for the extended capacitated arc routing problem. *IEEE Trans. Syst., Man Cybern. Part B. Cybern.*, **41**, 1110–1123.

YANG, S. (2003) Non-stationary problems optimization using the primal-dual genetic algorithm. *Proceedings of the 2003 IEEE Congress on Evolutionary Computation, CEC'03*, vol. 3., Piscataway, NJ: IEEE Press, pp. 2246–2253.

YANG, S. (2004) Constructing dynamic test environments for genetic algorithms based on problem difficulty. *Proceedings of the 2004 IEEE Congress on Evolutionary Computation, CEC'04*, vol. 2., Piscataway, NJ: IEEE Press, pp. 1262–1269.

YANG, S. (2005) Memory-based immigrants for genetic algorithms in dynamic environments. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference, GECCO'05*, New York, NY: ACM, pp. 1115–1122.

YANG, S. (2006a) Associative memory scheme for genetic algorithms in dynamic environments. *EvoWorkshops 2006: Application of Evolutionary Computation*. Lecture Notes in Computer Science, vol. 3907, Berlin: Springer, pp. 788–799.

YANG, S. (2006b) A comparative study of immune system based genetic algorithms in dynamic environments. *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO'06*, New York, NY: ACM, pp. 1377–1384.

YANG, S. (2006c) On the design of diploid genetic algorithms for problem optimization in dynamic environments. *Proceedings of the 2006 IEEE Congress on Evolutionary Computation, CEC'06*, Piscataway, NJ: IEEE Press, pp. 1362–1369.

YANG, S. (2008) Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. *Evol. Comput.*, **16**, 385–416.

YANG, S., CHENG, H. & WANG, F. (2010) Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks. *IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev.*, **40**, 52–63.

YANG, S., ONG, Y.-S. & JIN, Y. (eds). (2007) *Evolutionary Computation in Dynamic and Uncertain Environments*. Berlin: Springer.

YANG, S. & YAO, X. (2003) Dual population-based incremental learning for problem optimization in dynamic environments. *Proceedings of the 7th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, pp. 49–56.

YANG, S. & YAO, X. (2005) Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comput.*, **9**, 815–834.

YANG, S. & YAO, X. (2008) Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans. Evol. Comput.*, **12**, 542–561.

YU, E. L. & SUGANTHAN, P. N. (2009) Evolutionary programming with ensemble of explicit memories for dynamic optimization. *Proceedings of the 2009 IEEE Congress on Evolutionary Computation, CEC'09*, pp. 431–438.