Published in Computers and Operations Research, 135: 105359. DOI: 10.1016/j.cor.2021.105359

A Computational Study on Ant Colony Optimization for the Traveling Salesman Problem with Dynamic Demands

Sabrina M. de Oliveira[a,c,*], Leonardo C. T. Bezerra[b], Thomas Stützle[a], Marco Dorigo[a], Elizabeth F. Wanner[c], Sérgio R. de Souza[c]

[a] *IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium*
[b] *IMD, Universidade Federal do Rio Grande do Norte (UFRN), Natal, Brazil*
[c] *Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, Brazil*

Abstract

Ant colony optimization (ACO) algorithms have originally been designed for static optimization problems, where the input data is known in advance and is not subject to changes over time. Later, the long term memory of ACO proved effective for reoptimization over environment changes when extended to deal with dynamic combinatorial optimization problems (DCOPs). Among the major proposals of this kind, several adaptations of ACO procedures to improve information reuse can be identified, as well as a population-based ACO algorithm (P-ACO) specifically designed for DCOPs. Indeed, P-ACO drew the attention of the research community due to its ability to faster process pheromone information, but the few works assessing the effectiveness of the adapted ACO procedures and also P-ACO are not enough to reach more general conclusions on the current state-of-the-art in ACO for dynamic optimization. In this work, we conduct an extensive experimental campaign to evaluate the most common ACO procedure adaptations identified in the literature, using as underlying algorithms the state-of-the-art in ACO for static optimization ($\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, $\mathcal{MM}$AS) and the most relevant ACO algorithm proposed for dynamic optimization (P-ACO). A variant of the traveling salesman problem with dynamic demands (DTSP) is used as test benchmark, similarly to most investigations on ACO for combinatorial optimization. Besides the carefully designed experimental setup we adopt, our work represents a significant contribution for, at least, three reasons. First, our work is the first to acknowledge that DCOPs require custom-configured parameter settings, and also the first to use automatic configuration tools for this task. Concretely, we show how the hypervolume indicator can be used to evaluate and configure the anytime behavior of algorithms for DCOPs. Second, we directly compare $\mathcal{MM}$AS and P-ACO, isolating local search as an experimental factor. While P-ACO proves indeed effective in the absence of local search, $\mathcal{MM}$AS is able to consistently outperform it when local search is adopted. Finally, we conduct an experimental investigation on the DCOP-specific components proposed for ACO, once again isolating local search. Results show that those components contribute very little to performance when algorithms are allowed to use local search, but are remarkably effective in its absence. In fact, coupled with DCOP components, $\mathcal{MM}$AS outperforms P-ACO for a large part of our experimental setup.

Keywords: Ant colony optimization, dynamic traveling salesman, automatic configuration, hypervolume indicator.

1. Introduction

Ant colony optimization (ACO) [1, 2] has played a central role over the past decades as a successful metaheuristic for combinatorial optimization problems (COPs). In ACO algorithms, artificial ants search the solution space stochastically, biased by (i) *a priori* problem-specific heuristic information, and (ii) *pheromone* information knowledge acquired on-the-fly as the algorithm tackles an instance of the given problem. This pheromone-based memory

*Corresponding author.
Email addresses: oliveira.sabrina@gmail.com (Sabrina M. de Oliveira), leobezerra@imd.ufrn.br (Leonardo C. T. Bezerra), stuetzle@ulb.ac.be (Thomas Stützle), mdorigo@ulb.ac.be (Marco Dorigo), efwanner@cefetmg.br (Elizabeth F. Wanner), sergio@dppg.cefet.br (Sérgio R. de Souza)

of ACO algorithms has proven well suited to static COPs [1, 2], where the input data do not change during the run of the algorithm or before a solution is executed. In this scenario, the knowledge acquired by ACO algorithms in the form of pheromone accumulates into a strong bias towards promising regions of the search space.

The successful applications of ACO to static COPs stirred the interest of researchers that investigate dynamic COPs (DCOPs) [3–7], where *dynamic problem components* are allowed to change at runtime without notice. In the context of routing problems, examples of real-world dynamic components range from objective functions [8] to instance data, such as production costs, travel distances [9–12] or supply demands [13–21]. In contrast to the highly advantageous use of a pheromone memory in static COPs, a dynamic scenario makes the long-term knowledge of ACO a mixed blessing. On one hand, changes to the problem data are expected to only affect a portion of its original definition, and so ACO algorithms could possibly reuse information learned before the changes to speed up the re-optimization cycle. On the other hand, if an ACO algorithm has already converged to a specific region of the solution space that after a change is no longer interesting or even feasible, the algorithm search might be delayed by the need to first forget its previous knowledge.

The most innovative ACO proposal targeting DCOPs is an algorithm meant to optimize the reuse of pheromone information after problem changes, which we will refer to as *pheromone transfer*. In more detail, the P-ACO algorithm [22, 23] proposed a one-shot pheromone evaporation approach with the aid of a solution archive, in which the pheromone contribution from a given solution in the archive is only removed when that solution leaves the archive. Besides efficiency, this archive-based pheromone memory of P-ACO directly regulates the contribution timespan from solutions found before problem changes, and hence this algorithm quickly rose to become the ACO reference for DCOPs. However innovative, the studies targeting ACO for DCOPs that directly compared P-ACO to other proposals left important questions unanswered. Perhaps the most important, those studies have not sufficiently clarified the role played by *local search* [16, 21, 24, 25] in this context. The efficacy of ACO algorithms proposed for static COPs heavily rely on the efficient use of local search operators and the pressure they add on convergence, improving both the search speed and the expected quality of the final solution found by algorithms. More importantly, studies in the context of static COPs have shown that the result of comparisons between algorithms is strongly dependent on whether or not they are allowed to use local search operators [2], a fact likely to hold also in the context of DCOPs.

Another important question that has been overlooked in the ACO literature targeting DCOPs is the role of the *experimental setup* [1, 2, 26, 27] which includes (i) the proper configuration of parameters settings, and (ii) an adequate solution quality measurement. Concerning the former, the efficacy of ACO algorithms depends on balancing heuristic and pheromone information. This balancing is regulated by numerical parameters, and their proper setting requires significant knowledge of parameter configuration, ACO algorithms, and of the problem instances one is dealing with. Regarding solution quality measurement in dynamic optimization [11, 15, 19, 25, 27–33], many measures evaluate algorithms based solely on the quality of the final solution they produce, completely disregarding the performance of the algorithm during different re-optimization cycles. Among the measures that assess the behavior of algorithms over dynamic changes and solution quality development, some depend on a combination of other metrics [16, 21, 22], whereas others were proposed in the context of artificially designed test problems, where optimal solutions are known beforehand [11, 15, 19, 27].

In this work, we conduct a computational study to investigate the role of pheromone transfer, local search, and experimental setup in the performance of high-performing ACO algorithms for DCOPs. Our computational study is conducted on the traveling salesman problem (TSP) with dynamic demands [13, 16, 34]. We select the TSP because it is an extensively studied COP that has been used as test benchmark for many algorithms, and because ACO algorithms have been successfully applied to it. In fact, we observe a similar pattern in the context of DCOPs, with many proposals (ACO or not) being firstly assessed on this problem [1, 14, 17, 34, 35]. To assess algorithms on this variant of the TSP, we define a complete experimental design that is applicable to DCOPs in general. The most important component of the proposed design is an effective way to configure parameter settings and evaluate performance, which builds upon our preliminary work on the *anytime behavior* of dynamic optimization algorithms [34]. In that work, we have showed that the hypervolume indicator [36] does not present the disadvantages observed for existing quality measurements applied to dynamic optimization, and has additional advantages demonstrated in the multi-objective optimization literature. Here, we go one step further and show how it can be used to optimize the anytime behavior of dynamic optimization algorithms through *automatic parameter configuration* [36].

The investigation we conduct in this work is split into two stages. In the first stage, we compare P-ACO with $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS [37]), one of the best-performing ACO algorithms from static optimization. In

fact, its performance on static COPs motivated a number of adaptations to extend this successful algorithm to deal with DCOPs [11, 16, 18, 21, 25, 27]. We compare P-ACO and $\mathcal{MM}$AS (i) in their version configured for the TSP with dynamic demands versus the default configurations typically adopted for static and dynamic problems in general, and; (ii) with and without local search procedures. With (i), we demonstrate that a proper configuration of the ACO algorithms parameter settings can improve their performance, whether they have been proposed for static ($\mathcal{MM}$AS) or dynamic (P-ACO) optimization. With (ii), we demonstrate that using or not local search can drastically change the results of the comparison. In more detail, when local search procedures are not adopted, the computation time saved by the archive-based pheromone memory of P-ACO is impressive in comparison to $\mathcal{MM}$AS. In addition, most of the procedures related to pheromone transfer contribute to the efficacy of P-ACO. Conversely, when local search is introduced, the speed-ups from P-ACO become irrelevant since (i) local search requires a significant amount of time and (ii) $\mathcal{MM}$AS presents speed-ups of its own for dealing with local search. Under this setup, the performance of $\mathcal{MM}$AS significantly surpasses the performance of P-ACO.

The second stage of our investigation revisits some of the proposals to extend $\mathcal{MM}$AS to DCOPs [11, 16, 18, 21, 25, 27], assessing under our setup the improvements to $\mathcal{MM}$AS performance provided by those pheromone transfer mechanisms. More importantly, we use $\mathcal{MM}$AS as ACO test benchmark to understand if those components contribute to the performance of effective ACO algorithms in general. Among the most relevant insights we observe, $\mathcal{MM}$AS is able to outperform P-ACO for a large extent of the experimental setup we consider when coupled with pheromone transfer approaches. Yet, the benefits of those approaches become minimal in the presence of local search. Altogether, these results reinforce the need for computational studies that consider multiple real-world factors.

The main contributions of our work can be summarized as follows:

I. the definition of a complete experimental design for DCOP that includes the use of an effective way to define algorithmic parameter settings and evaluate performance;

II. the use of the hypervolume indicator, a well-known performance measure in the multi-objective research community, to assess the anytime behavior of DCOP algorithms;

III. a comparison of the most relevant ACO algorithms for static and dynamic combinatorial optimization;

IV. a computational study addressing the effectiveness of ACO procedure adaptations proposed in the dynamic combinatorial optimization literature.

The remainder of the paper is organized as follows. In Section 2, we introduce the existing dynamic variants of the TSP, highlighting the variant that we use as benchmark in this work. Section 3 gives an overview of ACO algorithms and their adaptations to deal with DCOPs. In addition, we also discuss the factors that have been overlooked in other investigations, namely local search procedures and parameter configuration. In Section 4, we target the performance assessment of DCOP algorithms, briefly reviewing the two major metric categories from a critical perspective, and detailing our approach to the automatic configuration in dynamic optimization. Section 5 describes our experimental setup, whereas the comparison of P-ACO and $\mathcal{MM}$AS is discussed in Section 6. Section 7 reports the computational study of the ACO adaptations for dynamic optimization. Finally, conclusions and future work are discussed in Section 8.

2. The dynamic traveling salesman problem

The *traveling salesman problem* (TSP) is one of the most studied NP-hard combinatorial optimization problems. It is a problem on which a number of important algorithmic ideas have been tested for the first time and it is also the problem to which ant system, the first ACO algorithm, was initially applied [38]. Since then, the TSP has been used frequently in ACO research to test new ACO algorithms and to evaluate their progress over ant system [1, 2].

The TSP is motivated by the task of a salesman that, from his hometown, needs to find a shortest tour through all the cities scheduled for a visit, and then back home. It can be represented by an undirected weighted graph, $G(V, E)$, where V is the set of $n = |V|$ vertices and E is the set of edges that fully connects the vertices. A weight d_{ij} is assigned to every edge (i, j). Here we assume that for all pairs of vertices i, j we have $d_{ij} = d_{ji}$, that is, the TSP is symmetric. The objective of the TSP is to find a Hamiltonian tour of minimum weight, where the weight of a tour is computed as the sum of all the weights of the edges crossed in it.

The *dynamic traveling salesman problem* (DTSP) is a variation of the TSP in which the problem data change over time. It has been proposed as an idealized model of time-varying problems that arise in many areas such as routing, logistics or production scheduling, and that make a problem more challenging. Two main variants of the DTSP can be found in the literature. The first variant considers that the distances between the vertices change over time, simulating the occurrence of traffic jams, accidents or the change of weather conditions [9, 11, 17]. The second variant considers dynamic demands, where the vertices that need to be visited in a tour change over time due to the cancellation of known visits or the appearance of new ones [13, 16, 18–21]. In this paper, we study this latter version, which can be modeled by a sequence of graphs $G_s = (V_s, E_s)$, $s = 0, ..., S$, and two sequences of vertex sets A_s and D_s, $s = 1, ..., S - 1$. In particular, A_s represents the set of new customers to be served and D_s represents the set of deleted customers. We then have that $G_0 = (V_0, E_0)$ is the starting graph and each V_s is obtained by $(V_{s-1} \bigcup A_s)/D_s$ and $E_s = V_s \times V_s$.

One important issue that arises for DOPs is to build up a proper test environment. Different ways have been proposed in the literature depending on how re-optimization is done over the run and whether changes are periodic, continuous, or cyclic. A review about several DOPs benchmark generators for both continuous and combinatorial DOPs is presented in [39, 40]. For our experiments here, we have used the same principle proposed in [14], where the dynamic environment is generated as follows. First, the set of customers from the actual problem instance is split into two sets called currentpool and sparepool, where the former defines the current problem instance to be tackled. At specific moments, a fraction of the vertices are switched between currentpool and sparepool to define the new problem instance, regulated by parameter $\xi \in [0, 1]$, here called *degree of dynamism*.

A second parameter that characterizes the instances is the *frequency of change*, which defines how often the instance in dynamic problem changes. While in [16, 18–21] the number of iterations/evaluations between changes was fixed, in this paper the periodic re-optimization occurs according to runtime. Our rationale is that real-world dynamic problems are subject to asynchronous changes in time, with no regard to algorithmic concepts such as the number of iterations or the use of function evaluations. However, for simplicity we keep the frequency of change fixed, i.e., environment changes happen synchronously producing f time intervals evenly split. Besides modeling real-world problems more accurately, this change in formulation has a direct impact in how algorithms should be engineered. More precisely, an algorithm that presents a high computational overhead at each iteration might have very few iterations to re-optimize solutions between changes, which is clearly an undesirable behavior. In the next section, we discuss how different ACO algorithmic components have been proposed to handle these characteristics of DCOPs.

3. Ant colony optimization

Ant system (AS) was the first ACO algorithm and was developed for solving the static TSP [38]. After that, many successful ACO variants have been developed and applied to a large number of COPs [1]. In general, ACO algorithms consist in a set of ants that build solutions biased by so-called (artificial) pheromone trails and heuristic information. The relative influence each one has in the solution construction is defined by two numerical parameters α and β, respectively. The pheromone trails encode a long-term memory about the search process of the ants. Effectively, this memory enables the colony of ants to reuse knowledge from solutions they have generated in previous iterations. The way pheromone update is implemented differs across ACO variants, and the choice of an appropriate pheromone update mechanism is essential to obtain effective ACO algorithms. Pheromone update usually involves pheromone evaporation, which is implemented as the reduction of the pheromone trail strength, normally by a fixed percentage, and the deposit of pheromone by one or several ants. A detailed description of pheromone update procedures of ACO algorithms can be found in [1].

This reusable source of information has raised the interest of researchers for applying ACO algorithms to DCOPs, since the pheromone matrix can be seen as a soft memory of solution components that have shown to be useful in the previous stage and may still be useful after environmental changes. To ensure the effectiveness of the learning procedure of the ants, however, one must decide how this past information should be considered in different dynamic environments, and many adaptations of ACO algorithms for DCOPs have been proposed in the literature [9, 13, 14]. In the following, we review the $\mathcal{MM}$AS [37, 41] and P-ACO [13, 22, 23] algorithms, two of the most relevant ACO algorithms that we will use in this paper. Next, we discuss the most important adaptation aspects required for

applying ACO algorithms to DCOPs such as the DTSP and, in particular, we consider the adaptation proposals from the literature involving $\mathcal{MM}$AS and P-ACO.

3.1. Underlying ACO algorithms for DTSP

$\mathcal{MM}$AS is one of the most efficient ACO algorithms for COPs, in particular the TSP [2, 37, 41]. As a consequence, it has been used as underlying algorithm to compare the performance adaptations of the ACO mechanisms for the DTSP [13, 22, 23, 42]. $\mathcal{MM}$AS handles pheromone trails concerning four major aspects: (i) the range of the pheromone trail strengths is bounded by numerical parameters τ_{max} and τ_{min}, that is $\forall \tau_{ij}, \tau_{min} \leq \tau_{ij} \leq \tau_{max}$; (ii) when the algorithm first starts, the pheromone values are initialized to τ_{max}; (iii) the pheromone trails are reinitialized every time the algorithm shows stagnation behavior, and; (iv) after each iteration, only one ant is allowed to add pheromone, which can be the best so far ant, T^{gb}, or the restart-best, T^{rb}. In addition, the pheromone update rule of $\mathcal{MM}$AS is given below,

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \triangle \tau_{ij}^{best} \tag{1}$$

where $\rho \in [0, 1]$ is the evaporation rate, $\triangle \tau_{ij}^{best} = 1/L^{best}$ if the edge (i, j) belongs to the best tour or 0 otherwise, and L^{best} is the length of the best tour.

We remark that, in the context of the TSP, many ACO algorithms such as $\mathcal{MM}$AS use nearest neighbor lists for speeding up solution construction. Concretely, whenever a DCOP moves from G_s to G_{s+1}, the set of customers to be attended changes. Thus, the best solution found so far and also the nearest neighbor lists may change or even be not valid anymore. In this paper, every time there is a change on G_s, the nearest neighbor lists are generated again for the new instance set defined,[1] and used to identify the best-so-far solution by using the nearest-neighbor heuristic.

P-ACO differs from most ACO algorithms in its way to manage pheromone. Instead of accumulating pheromone information provided by solutions constructed in every iteration, only solutions maintained in an archive are reflected in the pheromone matrix. Concretely, when a solution k enters the solution archive P (bounded to a maximum size K), pheromone is added on the edges it contains. When $|P| = K$, every new solution replaces another solution in the solution archive. When this replacement happens, the pheromone contribution of the components of the solution exiting P are removed.[2] The general form of the P-ACO pheromone matrix is as follows:

$$\tau_{ij}(t+1) = \tau_0 + \Delta \cdot |\{\pi \in P | (i,j) \in \pi\}| \tag{2}$$

$$\Delta = \frac{\tau_{max} - \tau_0}{K} \tag{3}$$

that is, the pheromone over edge (i, j) at a given iteration $t+1$ equals the initial pheromone deposit τ_0 plus a deposit of Δ, as many times as the number of occurrences of that edge in the solutions that belong to the archive.[3]

The approach of P-ACO to pheromone management has two major consequences. First, the pheromone update mechanism of P-ACO is typically faster than that of other ACO algorithms, which makes P-ACO a promising algorithm to be applied to dynamic COPs. Second, since only solutions in the archive influence the pheromone matrix, the archive management strategy is critical for the performance of P-ACO. Different archive management strategies were proposed by Guntsch [23] and their choice depends on the problem tackled. For the DTSP, the *age-based* strategy proved to be better or as competitive as the others proposed strategies [23], and hence it is the strategy adopted in later studies using P-ACO on DTSPs [42] and also here. Effectively, this strategy follows a FIFO *(first in first out)* queue behavior, where new solutions replace the firstly added to the archive.

[1] Note that the pheromone level of edges $V_{s-1} \setminus D_s \times A_s$ and $A_s \times A_s$, unless said otherwise, are initialized to τ_{max} and the other edges (that is, all those in $V_{s-1} \setminus D_s \times D_{s-1} \setminus D_s$) keep the same values as before the move.

[2] Note that, when the algorithm starts, all entries of the pheromone matrix have an initial value of τ_0, which has the same effect as the minimum pheromone trail limit in $\mathcal{MM}$AS [37].

[3] Note that the pheromone over edge (i, j) at a given iteration $t+1$ does not depend on the pheromone over that edge on previous iterations, and that the deposit does not depend on solution quality.

3.2. Adapting ACO to the DTSP

ACO algorithms have been originally designed for static COPs, with the goal of converging to a given region of high-performing solutions. As a result, if the pheromone information has already converged, the high levels of pheromone in most high-performing ACO algorithms will force the colony to follow a single trail, even after a dynamic change. To address this issue, different adaptations to maintain cross-environment search diversity or, more generally, improve the effectiveness of ACO algorithms in the context of dynamic optimization, have been proposed. In this section, we present an overview of the most relevant contributions found in the literature.

3.2.1. Pheromone transfer

The major proposals to improve ACO algorithms for DCOPs concern transferring pheromone information across different environments, typically by means of adaptations to the pheromone update procedure. Effectively, the pheromone transfer mechanism values the importance of the pheromone deposits from previous environments.

Guntsch [23] originally proposed that the pheromone information of the edges added and deleted should be (re)set to τ_0, with the resetting of the pheromone for edges V_{s-1}/D_s being regulated by a forgetting parameter $\gamma \in [0, 1]$, i.e., $\forall j \neq i, \tau_{ij} = (1-\gamma)\cdot\tau_{ij}+\gamma\cdot\tau_0$. Also concerning edges V_{s-1}/D_s, authors proposed a second approach comprising local assignments, where pheromone update values are computed based on the heuristic or the pheromone information, depending on the strategy adopted by the ACO designer[4]. Alternative update proposals are described in [43]. The first is dubbed local random restart, where the pheromone information of the newly added edges is uniformly randomly initialized a to value $r \in [0, 1]$. The second approach, dubbed local restart, zeroes the pheromone level for all newly added edges. However, authors remark that both approaches are only effective when combined with hyper-populations, and that otherwise the approaches proposed in [23] prove better.

Another set of approaches to pheromone transfer concerns immigrant schemes, firstly introduced in the context of evolutionary algorithms [28, 44–46], and later extended to ACO algorithms [11, 16, 19, 27, 47]. In general, an immigrant is either an ant or a solution that is moved from its original search context to another to promote diversification in a biased way. In dynamic ACO algorithms, this is a default practice as the pheromone information from many solution components created in past environments are reused in new ones. However, few proposals can be seen in the literature where randomly modified solutions are used as immigrants [15], thus introducing a perturbation to the pheromone that is transferred across environments.

An alternative immigrant-based scheme for pheromone transfer involves both pheromone deposit and evaporation [27]. In more detail, authors proposed a multi-colony version of $\mathcal{MM}$AS where colonies use different evaporation rates ρ, empirically defined, and after environmental changes they can exchange the best solution found so far by each colony. By migrating solutions, the pheromone structure of the colonies can be guided towards regions different from the ones colonies had converged to, and hence diversity is introduced. Later, a self-adaptive evaporation rate was proposed [20], where the value of the evaporation rate can be increased or decreased during the run of the algorithm to manage the influence of the knowledge transferred from previous environments.

Another multi-colony approach is proposed in [17], where ants are grouped in *castes* and attempt to re-optimize solutions with different exploration versus exploitation trade-offs. In more detail, castes adopt the *pseudo-random constructive rule* originally proposed for Ant Colony System (ACS [1]), differing as to the values adopted for its associated numerical parameter q_0. Thus, depending on how q_0 is set, some castes will search favoring exploitation, while other castes favor exploration. Although this variant does not alter pheromone values when transitioning between environments, it allows solution construction to value pheromone information less than the traditional ACO approach.

Recently, Mavrovouniotis *et al*. [5] reviewed the above approaches in the context of another DTSP variant, namely the TSP with dynamic weights. Though their conclusions escape the scope of this paper, we refer the reader to that work for further reference on pheromone transfer approaches.

3.2.2. Local Search for the DTSP

Local search (LS) procedures play an important role for ACO effectiveness on static COPs, providing the algorithm a means to explore locally a neighborhood in the search space. Among the most notable neighborhood operators

[4] We refer to the original work for further information on these options.

adopted in adaptations to the DTSP, we highlight the inver-over (IO) operator [48][5], the 2- and 3-opt operators [1], and the unstringing and stringing (US) operator [21]. All these operators plus the Lin-Kerninghan (LK) heuristic have been assessed in the context of the TSP with dynamic weights in [25, 32]. Once again, conclusions on a different DTSP variant do not fit our overview, but the reader is referred to those works for further reference on LS procedures used in ACO algorithms for DTSP problems.

From the experiments reported in the literature for the TSP with dynamic demands, target of our overview, the contribution of applying local search is clear in terms of solution quality improvement. On the other hand, when tackling dynamic problems such as the DTSPs, the time spent to improve solution quality is as important as the performance improvement itself, and so LS operators should be assessed having this trade-off in mind. Finally, another important remark concerning LS operators is that different algorithms can benefit from them in different rates, and hence a comparison that considers only algorithms with local search and another that considers only algorithms without local search may reach very different conclusions, as in the case of the static TSP [2].

3.2.3. Parameter settings

ACO algorithms are highly sensible to their parameter settings, as they regulate various aspects of their search behavior (e.g., the importance of pheromone versus heuristic information). Clearly, the change from static to dynamic environments defines a new problem class that likely requires parameter settings different from the default ones employed in static optimization. Two major groups of approaches can be identified in the literature concerning parameter settings. The first one considers multiple parameter settings within a single run of an algorithm, an approach modeled in ACO algorithms through the use of multiple species, castes, or, more generally, ant groups [17, 20].[6] The rationale behind these *multi-settings* approaches is that the features a dynamic problem may present, such as degree or frequency of change, might demand different trade-offs between exploration and exploitation, for instance. This way, by splitting the available computational effort among different search strategies, one expects to maximize the chances of performing effectively across a wide range of problem features. On the other hand, this increased robustness may cost the algorithm a significant share of its computational resources, compromising the overall gain.

The second major group of parameter-related approaches concerns configuration, whether *online* or *offline*. Approaches of the former type consider adaptive strategies that modify parameter configurations as a function of the environment changes, an approach also called parameter control [49]. A major disadvantage of online approaches is that the mechanisms used are typically limited towards adapting very few, often only one parameter. In addition, online approaches naturally introduce additional parameters that are used to regulate the adaptive strategies. More importantly, in the case of random environment changes like in this work, it is difficult to anticipate how parameters should be adapted on-the-fly.

In the second class of approaches, *offline configuration*, algorithms are configured on a training benchmark set before being actually deployed. Several manual approaches to offline configuration can be identified in the dynamic optimization literature [17, 20, 23, 27], relying on the well-known trial-and-error method that configures parameters sequentially. The major drawback of this approach is that it does not take into account that parameters in optimization algorithms often interact, e.g. the regulation of the pheromone and heuristic information of the ACO construction procedure. Alternatively (as we do in this work), automatic approaches [50–52] model parameter configuration as an heuristic optimization problem, where parameters are variables to be configured. The goal of the configurator is then to optimize the performance of the target algorithm according to a given metric, such as solution quality, runtime or the number of solutions constructed. In this work, we use an off-the-shelf automatic configurator (*irace* [52]) to fine-tune the parameter settings of the algorithms investigated, ensuring that our conclusions concern high-performing settings of the given algorithms.

In the next section, we discuss the metrics traditionally employed in the context of dynamic optimization to evaluate the performance of algorithms, and detail and motivate the metric we adopt for configuration and analysis.

[5] Although the authors do not consider the IO as a type of LS, the IO operator is a 2-opt exchange move without taking into account the heuristic information. In fact, an issue that arises from the experiments conducted in [48] is that comparisons are made only with algorithms that do not use LS, providing an unfair advantage to the IO-based algorithm.

[6] In fact, the critical feature that distinguishes these multi-settings approaches from the multi-colony approach of [27] is that multi-colony approaches consider multiple pheromone structures, whereas multi-caste or multi-species approaches use a single pheromone structure for all ant groups.

4. Improving performance via configuration in DCOPs

The performance analysis of stochastic optimization algorithms applied to DCOPs is a challenging, still maturing research area. As such, many performance metrics have been proposed, typically classified either as quality-based or behavior-based [29, 30, 42, 53]. In this section, we give a high-level overview of each category. Initially, we differentiate them between the ones adapted from static optimization and the ones proposed specifically for dynamic optimization. Later, we deepen our discussion of the hypervolume indicator, the primary metric of interest in our investigation, and explain how we use it for the automatic configuration of dynamic optimization algorithms.

4.1. Overview of metrics applied to DCOPs

The most common performance metrics for static COPs are based on the solution quality of the best solution found up to a given termination criterion (the final solution), averaged over a series of runs. For example, when the optimal solution of a given instance is known a priori, as with the most commonly used TSP instances [54], it is possible to compute the relative percentage deviation (RPD) between the best solution found by an algorithm and this optimal solution. In fact, the average of the RPD over a series of runs is likely the most commonly adopted metric in static optimization [55]. By contrast, the environment changes in dynamic COPs may require re-evaluations of the quality of solutions, due to changes to the input data and/or to the optimal solution [30, 56].

Many proposals adapting solution quality analysis for dynamic problems can be found in the literature [11, 19, 21, 25, 27, 28, 30, 39, 42, 53, 57–59]. Most of the proposals based on solution quality for DCOPs average the quality of the best solution found in each environment (the environment-final solutions) and are hence called *quality-based*. Conversely, *behavior-based* metrics provide a more comprehensive perspective of the performance of algorithms. In particular, the main issue with quality-based metrics is that different algorithms can display the same average values at the end of a run or period, but completely diverge on their search dynamics. In fact, the goal of a dynamic optimizer is not only to retrieve a high-performing solution at the end of its run (or over all environment changes), but also to proceed with this retrieval in an efficient way. In a sense, each environment change forces the algorithm to restart its search, and each of these optimization cycles must be able to retrieve high-performing solutions consuming as few resources[7] as possible.

4.2. Assessing the anytime behavior of a dynamic optimizer

In the context of static optimization, assessing the performance of an algorithm considering both resource consumption and solution quality is known as assessing its *anytime behavior* [36]. When the resource consumption to be minimized is runtime, solution quality over time (SQT) plots can be used for a graphical analysis [55]. For dynamic optimization, several behavior-based metrics extend analytically the SQT to deal with multiple environments [16, 29, 39, 58, 60, 61]. However, many such metrics have been proposed in a context where artificial benchmark problems are used, having the knowledge of the optimal solutions for each environment at hand. In most real-world situations (and also here), such metrics cannot be applied as the optimal solution is in constant change across the different environments (see Section 2).

Contrarily to this pattern, the *area between curves* (ABC) [62] is a behavior-based metric that does not require any assumptions about optimal solutions. This is illustrated in Figure 1, which depicts the performance fronts of two dynamic optimizers (left-most plots). In all plots, runtime is given on the x-axis, while solution quality is given on the y-axis (w.l.o.g. we consider a solution quality minimization problem). In more detail, the performance of a given algorithm Ψ_A is represented as a set of points $\Phi_A = (\langle \phi_A^{t_1}, t_1 \rangle, \langle \phi_A^{t_2}, t_2 \rangle, \ldots, \langle \phi_A^{t_{max}}, t_{max} \rangle)$, where $\phi_A^{t_i}$ is the solution quality of the best solution found by Ψ_A up to time instant t_i. The ABC metric computes the area between the performance fronts (or curves) of two algorithms, illustrated in Figure 1 (right). In fact, this metric is a binary variant of a metric known in the context of multi-objective as the *unary hypervolume indicator* [36], one of the best-established metrics for the performance analysis of multi-objective optimizers [63]. Specifically, the ABC metric is a particular case of the binary hypervolume metric where the reference point is only weakly dominated by the front assessed, and hence conclusions drawn from it cannot be guaranteed Pareto-compliant [36]. More importantly, this poor choice of reference point (albeit implicit in the metric definition) means environment-final solutions may not be properly valued.

[7] E.g., computational time or function evaluations.

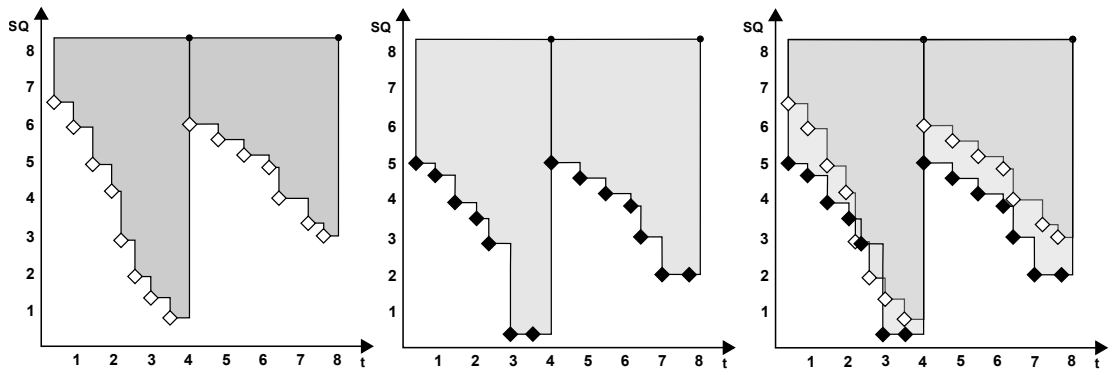


Figure 1: The hypervolume of two different sets of points (left and center), computed as a function of a bounding reference point. Given that points represent the performance of a dynamic optimizer, the hypervolume depicts its anytime behavior. In the dynamic optimization literature, the ABC metric (right) had been employed for a direct comparison of pairs of algorithms, measuring the difference between hypervolumes. Instead, in this work we compare multiple algorithms in an unary way, comparing the hypervolume values each algorithm achieves at a given run. We refer to the text for the reasons why we choose the unary hypervolume version over the binary one (ABC).

In a preliminary work, we have proposed the application of the hypervolume indicator to the context of dynamic optimization with minor adjustments [34]. The high-level idea is to compute the hypervolume dominated by each algorithm in each environment separately and aggregate over environment-wise hypervolumes to draw overall conclusions. As illustrated in Figure 1 (right), we adopt environment-wise hypervolume measurements since, if the whole run of an algorithm were considered as a single front, most of the points depicting a given environment would be considered dominated by the best solution of the previous environment. To ensure hypervolumes from multiple environments are comparable, we follow a two-stage approach. First, the values from both axes are globally scaled to ensure both axes always contribute equally to the hypervolumes. Second, the reference point for a given environment is computed as an x-axis translation of a global reference point. As a consequence, solutions from each environment need to be evaluated in isolation, since the reference point of a given environment would intersect with the next environment.

The most important advantage of the hypervolume over the ABC metric is the preservation of the benefits of the original anytime behavior formulation. Specifically, given two algorithms Ψ_A and Ψ_B, with their respective sets of performance-describing points Φ_A and Φ_B, the formally proven properties of the hypervolume indicator (I_H) ensure that, if Ψ_A presents a better anytime behavior than Ψ_B, then $I_H(\Phi_A) < I_H(\Phi_B)$. Alternatively, one can also say that, if $I_H(\Phi_A) < I_H(\Phi_B)$, then Ψ_A cannot present a worse anytime behavior than Ψ_B. Additionaly, using the unary hypervolume instead of its binary variants renders the analysis scalable w.r.t the number of algorithms compared. Regarding an overall analysis, the benefits of our approach vary as a function of the aggregation method considered. A rank sum analysis indicates how often one algorithm reacts more efficiently to problem changes than others. In the extreme case, an algorithm Ψ_A presents larger hypervolumes than another algorithm Ψ_B on all environments, so it is clear that Ψ_A cannot present worse anytime behavior than Ψ_B. Alternatively, one can also assess the average performance of an algorithm across environments. This flexibility of aggregation approaches is another improvement over the ABC metric, specially given that an algebraic sum implicitly embedded in the ABC metric provides less information than the alternatives discussed here.

We next describe how we use this formulation to enable the automatic configuration of dynamic optimizers.

4.3. Automatic configuration for dynamic optimization

As previously discussed, parameter settings are critical for the performance of metaheuristic-based algorithms such as ACO. Nonetheless, few are the works that conduct rigorous analysis of the parameter configuration best suited for dynamic optimization algorithms. More importantly, transferring parameter settings configured for one setup to be used in another setup is an approach prone to major drawbacks, as the literature on parameter configuration has repeatedly demonstrated [64]. In this context, automatic configuration tools can be instrumental, as they help address

the two major challenges regarding the configuration task, namely (i) the computational effort; and (ii) the expertise in parameter analysis.

In common, all offline configuration tools require four inputs: (i) a *parameter space*, i.e., the set of parameters to be configured and their respective domains, from which candidate configurations are sampled; (ii) a *training instance set* on which candidate configurations are run, different but representative of the *test instance set* for which algorithms are to be configured; (iii) the *performance metric* used to evaluate the candidate configurations on the training instances, and thus guide the configuration process, and; (iv) a pre-defined computational *configuration budget*, that limits the total number of experiments conducted. In the context of optimizing the anytime behavior of algorithms, the hypervolume metric is used as the configuration guiding metric [36]. As previously explained, the properties of the hypervolume ensure that optimizing this metric translates into optimizing the anytime behavior of the algorithm being tuned. In the next section, we present the experimental setup we adopt in our computational study, further detailing the configuration and testing setups we employ.

5. Experimental setup

Different adaptations of ACO algorithms and procedures for DCOPs have been proposed in the literature, but no study targeting the potential interactions between adaptation proposals has yet been conducted. In this work, we conduct such an investigation specifically targeting the three aforementioned sets of proposals, namely (i) pheromone transfer, (ii) local search, and (iii) parameter settings. We initially select an underlying ACO algorithm to use as baseline for comparison. In particular, we first conduct a comparison between P-ACO and $\mathcal{MM}$AS to understand which algorithm performs better in the context of the DTSP with dynamic demands studied in this work. Next, we create a set of variants of the selected baseline ACO algorithm, differing by the addition of a single pheromone transfer proposal. Effectively, the assessment of a variant is actually an assessment of how the algorithmic component that characterizes the given variant contributes to the performance of high-performing ACO algorithms in the context of DCOPs. In addition, to understand the importance of the other two experimental factors (local search and parameter settings), each experiment we conduct is performed with and without local search, and with different parameter settings that are obtained both from manual and automatic configuration.

Next, we detail the experimental setup we adopt, individually detailing benchmark, performance evaluation, configuration, and testing setups. We remark that all adaptations of ACO algorithms for the DTSP were implemented and executed on top of the ACOTSP software package available at `http://www.aco-metaheuristic.org/aco-code/`. Experiments conducted in this paper have been run on AMD Opteron CPUs with 12MB cache and 16 GB of RAM running under Cluster Rocks Linux. The algorithms studied are coded in `C` and compiled with `gcc` version 4.1.2.

Benchmark. Two sets of instances were used in the experiments conducted in this work, to further understand how algorithms perform when faced with instances that are structurally different. The first set is composed of instances taken from the TSPLIB instance benchmark [54], ranging from 1000 to 3000 cities[8]. The second set is composed of random uniform Euclidean (RUE) instances, with sizes depending on whether local search is used or not[9]. Dynamic environments were generated using the method described in [42] and also in Section 2. In particular, we consider currentpool and sparepool with equal sizes, i.e., each pool contains half of the cities from the original instance. As also explained in Section 2, two parameters define by how much instances change (degree of dynamism, ξ) and how often they change (frequency of change, f). In our experiments, f is set to 2 and 10, respectively meaning that changes occur after half and a tenth of the maximum allowed runtime. The degree of change is set to $\xi \in \{20\%, 40\%, 80\%\}$, and so the combinations of ξ and f comprise 6 different scenarios.

Evaluation. As previously discussed, algorithms applied to DCOPs have been traditionally compared using different performance metrics. In common, both final-quality based and behavior-based metrics have been indiscriminately assessed concerning number of iterations, function evaluations, or solutions generated [13, 22, 29, 30, 42, 53, 58,

[8] The selected TSPLIB instances are rl1323, u1817, rl1889, u2152, pr2392. The instance name gives the number of cities in the instance.

[9] Specifically, we use 2 subsets of 5 instances with sizes ranging from 2500 to 3000 when local search is not applied, and 3 subsets of 5 instances with sizes ranging from 3000 to 4000 when local search is applied. All RUE instances were generated using the portgen generator from the 8th DIMACS Implementation Challenge.

Table 1: Parameters space used as input for irace when configuring $\mathcal{MM}$AS and P-ACO. For brevity, we use the (x_i, x_n) notation to represent a discrete interval between x_i and x_n, where all integer values are considered by irace.

Algorithm	m	ρ	α	β	q_0	τ_{max}	K
$\mathcal{MM}$AS	(5, 100)	[0.1, 1]	[0, 5]	[1, 10]	–	–	–
P-ACO	(5, 100)	–	[0, 5]	[1, 10]	–	[1, 10]	[1, 25]

62, 65]. The rationale behind these resource consumption metrics is understanding by how much algorithms are able to improve solutions within a given iteration, of after a fixed number of function evaluations / solutions generated. Although they share the benefit of being hardware-independent (as long as no maximum runtime is established), they also mask the computational overhead of the algorithms, an effect that is particularly undesirable in the context of dynamic optimization. In this work, we evaluate algorithms using the hypervolume metric detailed in Section 4, considering as resource consumption metric the runtime of the algorithms. Concretely, algorithms are allowed a maximum runtime of 2 000 seconds when they use local search, and 1 000 seconds otherwise. By doing so, we put on evidence the most important characteristic algorithms designed tackling DCOPs should present, namely to be efficient as to the runtime they require for reoptimizing solutions.

Configuration. As detailed in Section 4, automatic algorithm configurators such as irace should be used within a carefully designed configuration setup. To meet this need, we formally define a configuration space given in Table 1 based on the ACO literature [1, 13, 22, 23, 42], delimiting parameters and domains for each algorithm we configure in this investigation.

Concerning the separation between configuration and testing benchmark sets, we created an alternative benchmark set for configuration, comprising thirteen instances ranging from 100 to 3000 cities taken from the TSPLIB instance benchmark [54][10], and 15 RUE instances, ranging from 2 000 to 4 000 cities. Additionally, to prevent floor effects that could reduce the effectiveness of the automatic configuration, when configuring an algorithm allowed to use local search we only adopt instances with more than 600 cities. Concerning the configuration budget, irace is given a maximum of 5 000 experiments for each configuration campaign. Candidates are evaluated according to the hypervolume metric, with reference points computed on-the-fly, and are discarded based on Friedman's non-parametric rank sum test using the default configurations of irace.

Testing. To account for the stochastic nature of ACO, each algorithm and their variants are executed 20 times on each instance. The qualitative analysis is conducted with the aid of solution quality over time (SQT) plots, depicting the average performance of each algorithm considered, measured according the hypervolume metric. The same averaged hypervolume approach is used to draw overall conclusions, in this case with the aid of Friedman's non-parametric rank sum test.

6. Comparing $\mathcal{MM}$AS and P-ACO

In this section, we discuss results from the first set of experiments we conducted. Overall, the goal of this first stage of the investigation is to compare the two best-performing ACO algorithms from the context of static and dynamic optimization, respectively $\mathcal{MM}$AS and P-ACO, isolating the effects of local search and parameter settings. Concretely, we run each algorithm with and without local search, using three different settings:

1. *Default*: the default settings used by $\mathcal{MM}$AS and P-ACO in the static COP literature [66, 67];
2. *Dynamic*: the settings used by these algorithms in the dynamic COP literature [21, 25, 32, 42], and;

[10]The selected TSPLIB instances are rd100, kroA150, kroB200, gr202, pr226, pr439, gr666, u724, vm1084, rl1304, vm1748, u2319 and pcb3038.

Table 2: Default (top) and configured (bottom) parameter settings used for $\mathcal{MM}$AS (settings with the *mmas* prefix) and P-ACO (settings with the *paco* prefix). The *LS* suffix is appended to settings used on experiments where local search is adopted. We remark that settings for *pacoDynamicLS* are not given since no study has yet investigated P-ACO coupled with local search for dynamic optimization.

	$\mathcal{MM}$AS + P-ACO			$\mathcal{MM}$AS	P-ACO	
Settings	m	α	β	ρ	τ_{max}	K
mmasDefault	$n/4$	1	2	0.2	–	–
mmasDefaultLS	25	1	2	0.2	–	–
mmasDynamic	50	1	5	0.2	–	–
mmasDynamicLS	50	1	5	0.2	–	–
pacoDefault	$n/4$	1	2	–	3	25
pacoDefaultLS	25	1	2	–	3	1
pacoDynamic	50	1	5	–	3	3
mmasTuned	96	1	5	0.6	–	–
mmasTunedLS	9	1	1	0.4	–	–
pacoTuned	79	1	3	–	3	25
pacoTunedLS	5	2	2	–	3	1

3. *Tuned*: a parameter setting that is obtained with automatic configuration for the context of the DTSP, as previously detailed.

Note that, in the first part of our investigation, neither of the ACO algorithms employ the *pseudo-random proportional rule* proposed by ACS [68], where a numerical parameter q_0 is used to regulate the balance between exploitation and exploration (this will be investigated in Section 7). In addition, parameter settings often have their local search counterparts, that is, parameter settings specific for the experiments where local search is adopted. We start this analysis with the insights concerning the effects of the automatic parameter configuration.

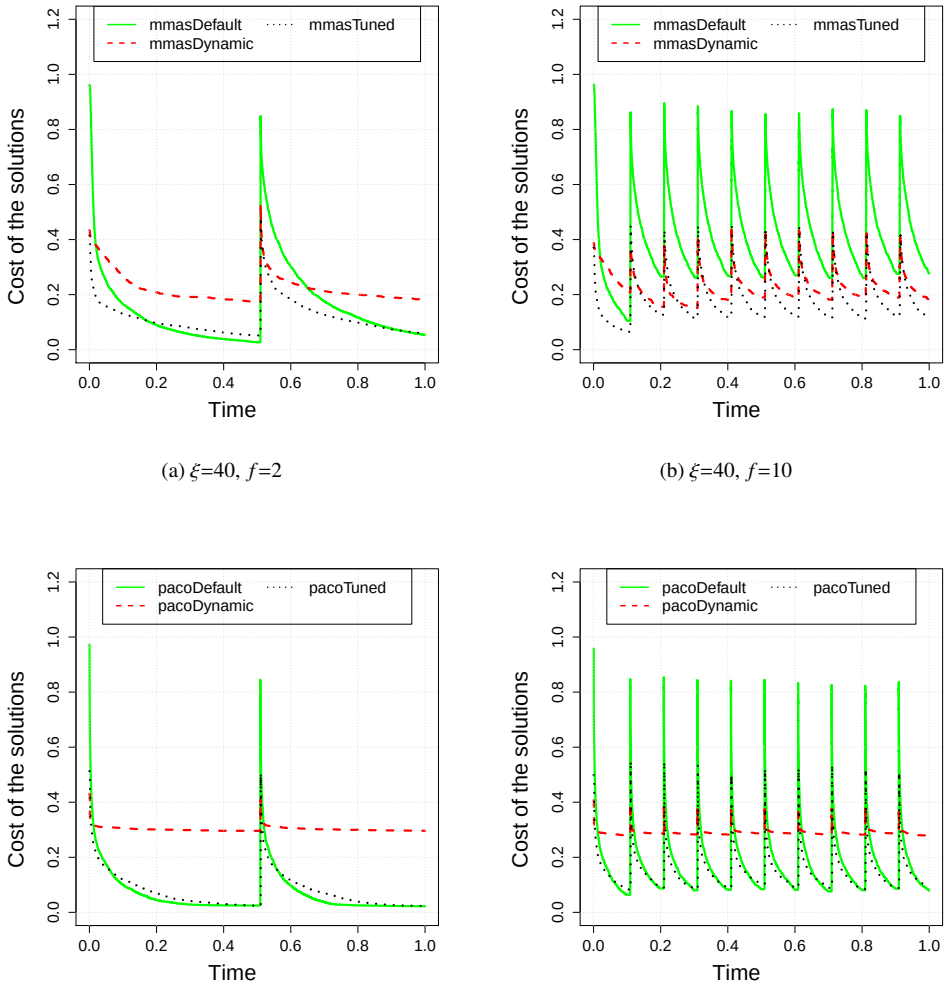
6.1. Assessing parameter settings

Parameter settings used in these experiments are given in Table 2, where default settings are given in the top rows, whereas configured settings are given in the bottom ones. In addition, the suffix *LS* is added to settings that are used in experiments where local search is adopted. Concerning default settings, a few observations stand out. First, despite their different structural characteristics, P-ACO and $\mathcal{MM}$AS have typically been run with the same parameter settings. Second, settings are very similar whether local search is used or not. Finally, the most noticeable difference between settings from the static and the dynamic optimization literature is the value of β, in an attempt to provide algorithms with stronger convergence pressure and thus speed-up solution re-optimization.

By contrast, the configured settings selected by irace given in Table 2 (bottom) are different for each algorithm, as well as between runs that use local search and runs that do not. Concerning $\mathcal{MM}$AS, for instance, we notice a larger value of ρ that makes the search more explorative. Effectively, this setting helps the algorithm escape from the search space region to where it had converged before an environment change, but had not yet been considered by manual configuration. The value of β changes as a function of local search. When local search is not adopted (*mmasTuned*), β is also increased, to allow the algorithm faster exploitation due to the limited amount of runtime available. Conversely, the strong exploitation nature of the local search component induced a decrement of β in *mmasTunedLS*.

Regarding P-ACO, parameters configured by irace for the experiments without local search (*pacoTuned*) greater resemble the settings adopted in the static optimization literature (*pacoDefault*) than the ones used in the dynamic optimization literature (*pacoDynamic)*. Interestingly, this resemblance between static and configured settings also holds in the presence of local search (*pacoDefaultLS* and *pacoTunedLS*). To empirically assess the impact of the different parameter settings presented in Table 2, we discuss below the most important insights observed with the help of SQT plots.

Dynamic characteristics of the scenarios. Figure 2 shows the anytime performance of $\mathcal{MM}$AS (top) and P-ACO (bottom) on a TSPLIB instance, run without local search using the three different parameter settings on two different experimental scenarios. In common, both scenarios present a degree of dynamism $\xi = 40\%$, but differ as to the frequency of change $f \in \{2, 10\}$. The overall pattern depicted in these plots shows how parameter settings can be affected by the dynamic characteristics of the scenarios. Notice, for instance, the anytime performance of $\mathcal{MM}$AS settings on the topmost plots. On the left (Figure 2a), the default settings of $\mathcal{MM}$AS (*mmasDefault*)



(a) ξ=40, f=2

(b) ξ=40, f=10

(c) ξ=40, f=2

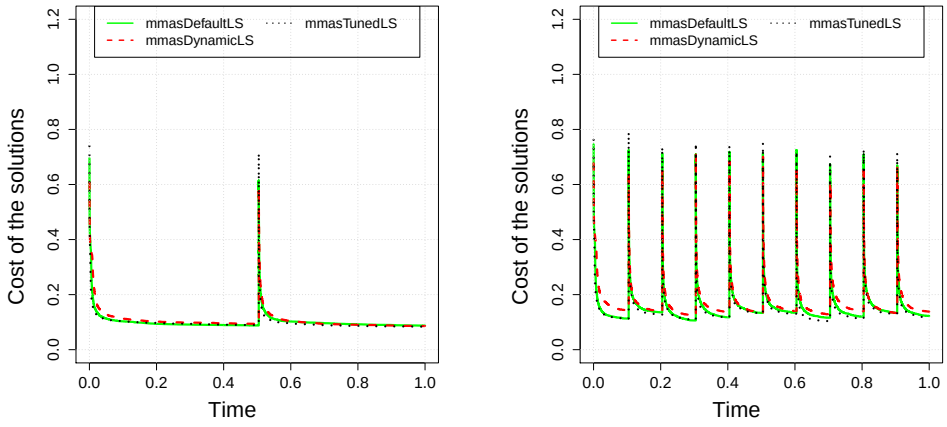
(d) ξ=40, f=10

Figure 2: SQT plot depicting the anytime performance of $\mathcal{MM}$AS (top) and P-ACO (bottom) on TSPLIB instance pr2392, without local search, run with different parameter settings on different dynamic scenarios.

improve over the dynamic settings (*mmasDynamic*), but the opposite happens on Figure 2b. In summary, an increment in the frequency of change is enough to alter the relative performance of the settings. The configured settings (*mmasTuned*) represent a compromise solution that shows robust performance across different scenarios.

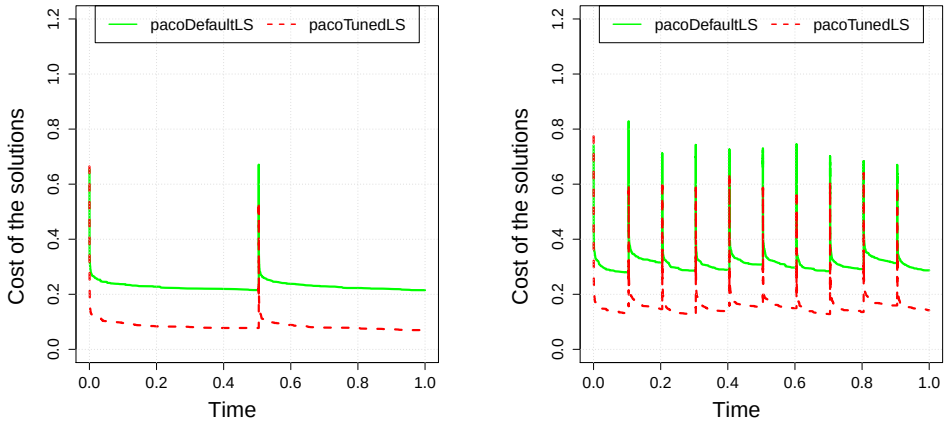
Regarding P-ACO, the improvements obtained with the help of automatic configuration are significant for most scenarios. Yet, for specific scenarios such as the one depicted in Figure 2c, P-ACO with default and configured configuration settings (*pacoDefault* and *pacoDynamic*, respectively) present similar performance. Moreover, we remark that the worst performance observed in Figures 2c and 2d concern P-ACO run with dynamic settings (*pacoDynamic*), which reinforces the importance of automatic parameter configuration methodologies.

Interactions with local search. Figure 3 shows the anytime performance of $\mathcal{MM}$AS and P-ACO when run with local search on the same instance and scenarios from the previous analysis. Although the conclusions about the effects of the dynamic characteristics of the scenarios still hold, the use of local search significantly changes the



(a) ξ=40, f=2

(b) ξ=40, f=10



(c) ξ=40, f=2

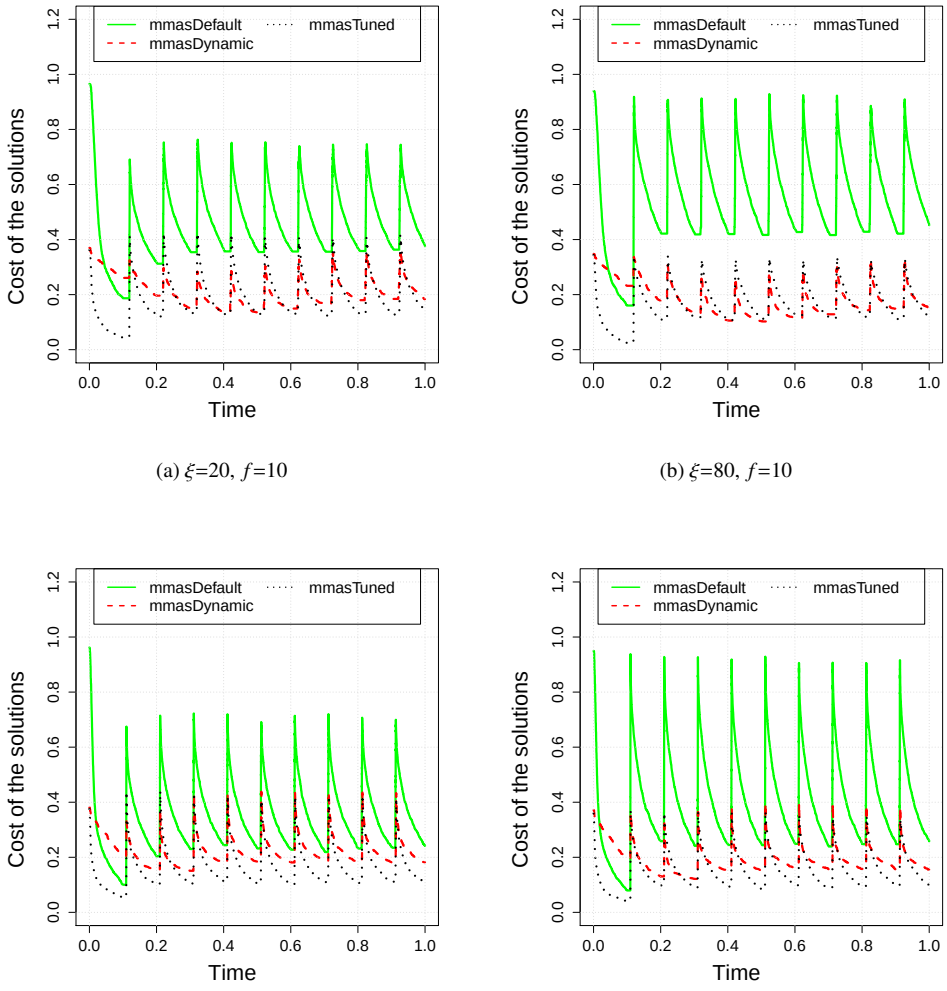
(d) ξ=40, f=10

Figure 3: SQT plot depicting the anytime performance of $\mathcal{MM}$AS (top) and P-ACO (bottom) on TSPLIB instance pr2392, with local search, run with different parameter settings on different dynamic scenarios.

improvement rate provided by automatic configuration for $\mathcal{MM}$AS. This is easily explained by the high-quality solutions obtained when local search is adopted, which makes any further improvement much more difficult to be obtained. The performance of P-ACO, on the other hand, is significantly improved by the configured settings. Furthermore, the benefits of automatic configuration are consistent through both scenarios.

Structural characteristics of the benchmark instances. The differences in benchmark instance characteristics are reflected in the performance displayed by different parameter settings, at least in the case of $\mathcal{MM}$AS. Figure 4 helps explain our observation, showing the performance improvements obtained from automatic configuration when running $\mathcal{MM}$AS on scenarios with $f = 10$ and $\xi = 20\%$ (left) or $\xi = 80\%$ (right). Specifically, results from RUE instances (top) show a much larger improvement provided by configuration, even if the improvement seen on the results from TSPLIB instances (bottom) is also significant.

A possible explanation is the importance of cities (customers to be attended) in the different benchmark sets. In more



(a) ξ=20, f=10

(b) ξ=80, f=10

(c) ξ=20, f=10

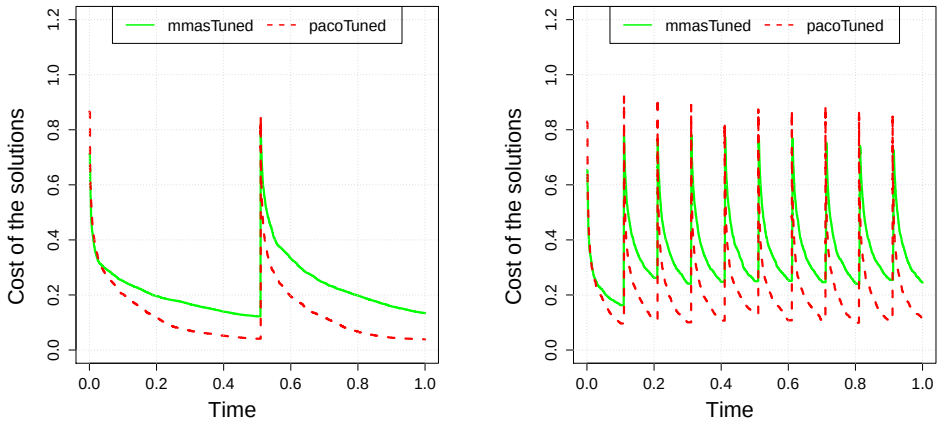
(d) ξ=80, f=10

Figure 4: SQT plots depicting the anytime performance of $\mathcal{MM}$AS on RUE instance 3002 (top) and on TSPLIB instance pr2392 (bottom), without local search, run with different parameter settings on different dynamic scenarios.

detail, RUE instances are expected to present cities of similar relevance to solution quality, whereas the relevance of TSPLIB instance cities may differ considerably. These results indicate that, though the configured settings are robust w.r.t. benchmark sets, an a priori knowledge of these features would likely benefit the configuration process.

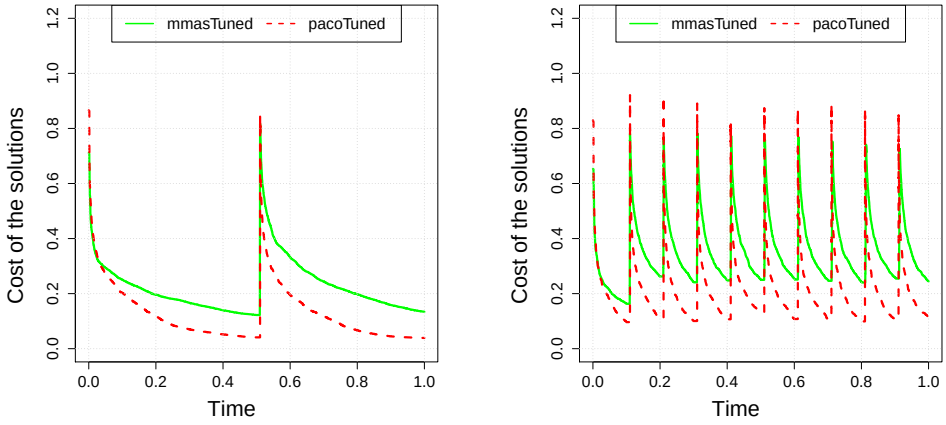
6.2. Performance comparison

We next directly compare $\mathcal{MM}$AS and P-ACO, once again isolating the effects of local search. For these experiments, however, we consider only the configured parameter settings, as the results from the previous section have confirmed that the automatic configuration methodology leads to improvements in the anytime performance of the algorithms. Overall, the insights we observe are considerably different for the experiments with and without local search, corroborating our claim that experimental analyses should isolate this factor. Figure 5 illustrates the most important insights we observe from each set of experiments, which we discuss below.



(a) ξ=40, f=2

(b) ξ=40, f=10



(c) ξ=40, f=2

(d) ξ=40, f=10

Figure 5: SQT plots depicting the anytime performance of P-ACO and $\mathcal{MM}$AS configured by irace, on TSPLIB instance pr2392 with (top) and without (bottom) local search run on different dynamic scenarios.

Experiments without local search. When local search is not adopted (top-most plots), the difference in performance between $\mathcal{MM}$AS and P-ACO across different environments is significant. More precisely, $\mathcal{MM}$AS demonstrates its best performance before the first environment change, whereas P-ACO displays a constantly good performance throughout the run. In addition, the effect for $\mathcal{MM}$AS is much weaker for $f = 2$ (left) scenarios than when $f = 10$ (right). Altogether, these observations can be justified by the characteristics of the ACO algorithms being compared. $\mathcal{MM}$AS is an algorithm originally designed for static, final-quality optimization, and thus it requires a given minimum runtime to produce good results. Moreover, the best applications of $\mathcal{MM}$AS rely on local search to increase its convergence pressure, an aid we specifically forbid in this set of experiments. Concerning P-ACO, its pheromone transfer mechanism and the fast update procedure appear to be the keys for its good performance. In addition, P-ACO studies have only started considering local search recently, and hence, the original algorithm has been designed to be effective without this extra source of pressure.

Table 3: Statistical analysis of $\mathcal{MM}$AS and P-ACO using different parameter configurations, without and with local search, aggregated over all instances and scenarios considered. Rank sum differences w.r.t. the best ranked algorithm are given in parenthesis.

Without LS	*Mean* (ΔR = 36.77)	***pacoTuned***	***pacoDefault*** (**28.5**)	*mmasTuned* (142)	*mmasDynamic* (257)	*mmasDefault* (313.5)	*pacoDynamic* (405)
With LS	*Mean* (ΔR = 9.3)	***mmasTunedLS***	*mmasDefaultLS* (110)	*mmasDynamicLS* (190)	*pacoTunedLS* (304)	*pacoDefaultLS* (406)	

Experiments with local search. Results change considerably when algorithms are allowed to use local search (bottom-most plots). In particular, both algorithms benefit from the additional convergence pressure, but $\mathcal{MM}$AS is now able to outperform P-ACO by a significant margin across all instance types, sizes, and scenarios. More importantly, $\mathcal{MM}$AS is no longer affected by the runtime available (nor, consequently, by the dynamic characteristics of the scenarios). Analogously, the performance of P-ACO is the same across all environments, whatever the instance type, size, or scenario considered. A deeper analysis revealed that the advantage of the faster pheromone update designed for P-ACO is much reduced, since relatively the local search procedure uses a significant amount of time. On the other hand, $\mathcal{MM}$AS presents speed-ups specifically conceived to improve the efficiency of its coupling with local search procedures, e.g., avoiding updating the pheromone matrix as a whole. Altogether, these changes in the relative efficiency of the algorithms explain the change in their anytime performance, with $\mathcal{MM}$AS clearly becoming the best option for the DTSP when local search is allowed.

6.3. Statistical analysis

To support the insights discussed in this section, Table 3 provides a rank sum analysis where we compare all variants of the algorithms, isolating the effect of local search. In more detail, we conduct three levels of aggregation. At the bottom level, we evaluate an algorithm w.r.t. to the average hypervolume it produces over all environments in a given run. At the mid-level, we evaluate an algorithm w.r.t. its average performance over the 20 runs on a given block, i.e., a given instance from a given experimental scenario. At the top level, for each block algorithms are ranked in ascending order according to their performance, and the sum of the ranks obtained by an algorithm on all blocks depicts its overall performance. These rank sums are used to assess statistical significance with 99% confidence level using Friedman's non-parametric test and associated post-hoc test [69]. The critical difference in ranks indicated by the test is provided as ΔR. Algorithms that present a rank sum difference w.r.t. to the best ranked smaller than ΔR are highlighted in boldface, indicating that no statistical significant difference was observed between the performance of the given algorithm and the best ranked one.

As discussed from the plots, when local search is not adopted, the performance of P-ACO is very similar both when default and configured settings are adopted. Ironically, the worst performance among all algorithms is observed for the settings typically adopted for P-ACO in the dynamic optimization literature. Regarding $\mathcal{MM}$AS, the configured settings improve over the manually-configured ones, but not enough to match the performance of P-ACO. Conversely, conclusions change completely when local search is adopted. For this setup, $\mathcal{MM}$AS is able to outperform P-ACO no matter the configuration adopted. More importantly, the configured version of $\mathcal{MM}$AS statistically significantly outperforms both the settings adopted in static and dynamic optimization. Regarding P-ACO, configured settings improve over the settings adopted in static optimization, but not enough to match the performance of $\mathcal{MM}$AS.

6.4. Concluding remarks

The experiments comparing the top-performing ACO algorithms from the context of static and dynamic optimization have confirmed that the most important factors in such an analysis are the appropriate configuration of parameters and the use of local search. In real-world scenarios, parameter configuration and local search procedures need to be assessed as to their practicality, since it may not always be the case that one is able to run the number of experiments required by offline configuration, or the expected time available for each environment can be too small for local search to be of any benefit. Nonetheless, it is easy to conceive a database of configured parameter settings for different problem benchmarks, scenarios, and stopping criteria, greatly reducing the overhead of offline configuration in practice. In the next section, we investigate the benefits of approaches proposed in the dynamic optimization literature to improve ACO performance.

Table 4: Parameters selected by irace for $\mathcal{MM}$AS variants using different pheromone transfer approaches. Settings specific to runs with local search are indicated by the *LS* suffix.

	$\mathcal{MM}$AS				*reset*	*multi*
Variant	m	α	β	ρ	γ	q_0
mmasTreset	96*	1*	5*	0.6*	0.8	–
mmasTresetLS	9*	1*	1*	0.4*	0.7	–
mmasResetT	95	1	5	0.4	0.9	–
mmasResetTLS	9	1	2	0.4	0.6	–
mmasRestartT	96	1	5	0.3	–	–
mmasRestartTLS	6	1	1	0.1	–	–
mmasMultiT	91	1	2	0.3	–	0.3
mmasMultiTLS	9	1	1	0.8	–	0.3
mmasMultiRestartT	98	1	5	0.4	–	0.4
mmasMultiResetT	95	1	5	0.3	0.4	0.2

* Settings reused from the experiments conducted in the previous section.

7. Assessing the effectiveness of pheromone transfer proposals

The experiments discussed in the previous section showed that, when properly configured and in the absence of local search, the pheromone transfer approach used by P-ACO is particularly effective. Conversely, since $\mathcal{MM}$AS was proposed for static optimization, it fails to display competitive performance after the first problem change. In this section, we assess the effectiveness of some of the pheromone transfer proposals reviewed in Section 3 when coupled with $\mathcal{MM}$AS. Initially, we investigate the benefits of adding to $\mathcal{MM}$AS the pheromone transfer approach from P-ACO, while isolating the effects of parameter configuration and local search. Next, we compare several pheromone transfer approaches previously discussed, once again isolating the effects of local search. Finally, we assess the combination of transfer approaches and compare the best-performing variant to P-ACO.

7.1. Assessing pheromone transfer through reset

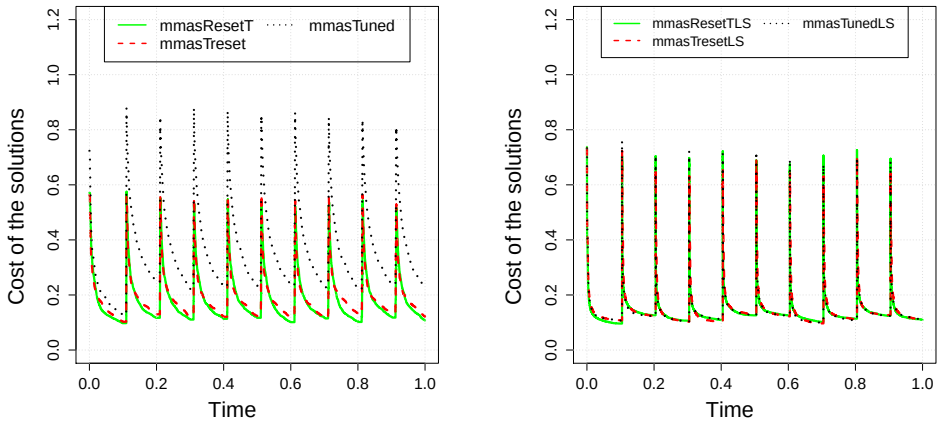
As discussed in Section 3, P-ACO originally proposed that the pheromone information on edges V_{s-1}/D_s should be (re)set to τ_0 after each environment change, regulated by a forgetting parameter $\gamma \in [0, 1]$. In the context of $\mathcal{MM}$AS, this translates into (re)setting the pheromone information to τ_{max}, since this is the initial pheromone value it adopts.

For the set of experiments conducted here, we compare three variants of $\mathcal{MM}$AS. The first version is the one we configured in the previous section (*mmasTuned*), which does not use any pheromone transfer mechanism, and serves as a baseline. The second variant uses the same parameter settings of the first one, but adopts the pheromone reset approach (*mmasTreset*).[11] Finally, the third version also adopts pheromone reset, but has been re-configured by irace (*mmasResetT*). Our rationale is that the addition of pheromone transfer mechanisms could lead to interactions with other algorithmic components of $\mathcal{MM}$AS and thus require a different parameter configuration. Configured settings for this set of experiments are given in Table 4 (top rows). Like before, experiments are run isolating the effects of local search, and the *LS* suffix is added to configurations where local search is adopted.

In the following, we discuss the most important insights we observe from our analysis, with the help of the SQT plots given in Figure 6. In particular, the plots depict the anytime performance of the three different $\mathcal{MM}$AS variants we consider when run on a TSPLIB instance with (left) and without (right) local search, on the $\xi = 40\%$, $f = 10$ scenario.

Parameter settings. The only parameters in which the settings of *mmasTreset* and *mmasResetT* differ are the evaporation (ρ) and the forgetting (γ) rates. The similarity also holds between the variants that use local search (*mmasTresetLS* and *mmasResetTLS*), though this time it is β rather than ρ that is changed by irace. From an algorithmic point of view, the most likely explanation for these changes concern the absence of a pheromone transfer mechanism in the original $\mathcal{MM}$AS, which was compensated by irace with an increased ρ value. Concerning performance, both $\mathcal{MM}$AS variants that adopt the pheromone transfer mechanism from P-ACO perform similarly, although differences are mostly observed in favor of *mmasResetT*.

[11] Since the pheromone reset introduces the forgetting parameter γ, we only configure this parameter.



(a) ξ=40, f=10

(b) ξ=40, f=10

Figure 6: SQT plots depicting the anytime performance of three different $\mathcal{MM}$AS versions, run on TSPLIB instance pr2392 with (right) and without (left) local search, using different parameter configurations on the $\xi = 40$, $f = 10$ scenario.

Improvements over the original $\mathcal{MM}$AS. Figure 6 (left) demonstrates the significant performance improvements provided by pheromone transfer through reset. In particular, the transfer approach helps address the most concerning issue with the original $\mathcal{MM}$AS, in that the performance of the variants are consistent throughout the run. More importantly, the benefits of transfer are observed for all instance sets, scenarios, and parameter configurations, as shown in the rank sum analysis given in Table 5. Conversely, the effects pheromone transfer mechanism are greatly reduced in the presence of local search, as shown in Figure 6 (right). In fact, it is often possible to identify performance differences in favor of the original $\mathcal{MM}$AS over the variants that adopt pheromone transfer. This is an important finding that corroborates our claim that algorithmic components proposed for dynamic optimization algorithms should be carefully assessed, in particular as to their possible interactions with other relevant components algorithms might use, such as local search.

7.2. Comparing pheromone transfer approaches

The insights produced in the previous section demonstrate that pheromone transfer approaches may interact with other algorithmic components, and hence reconfiguring variants is indicated. In this section, we compare three $\mathcal{MM}$AS variants, differing as to the transfer approach they adopt. The first variant is *mmasResetT*, which we preliminarily assessed in the previous section and serves as baseline. The second variant (*mmasRestartT*) restarts the pheromone trails of edges V_{s-1}/D_s to τ_{max}, completely forgetting the deposits from previous environments.[12] Finally, the third variant (*mmasMultiT*) follows the multi-caste approach [17] discussed in Section 3. In our work, ants are split into two groups of equal size $\lfloor m/2 \rfloor$. The first group of ants constructs solutions using probability q_0, whereas the second group uses probability $1 - q_0$. Thus, depending on how q_0 is set, half of the ants will search favoring exploitation, with the other half favoring exploration.

Parameter settings. Configured settings for this set of experiments are given in Table 4 (middle rows). The different pheromone transfer strategies produce little effect on the number of ants (m) and importance of the pheromone information (α). Indeed, the only noticeable differences between settings used by the variants concern the importance of heuristic information (β) and the evaporation rate (ρ). However, no overall pattern can be easily observed, although it is remarkable that the variants that use local search present so contrasting ρ values. In particular, the very low ρ

[12] Notice that the *restart* variant is equivalent to configuring parameter γ to its maximum value in the *reset* variant.

Table 5: Statistical analysis of the experiments conducted in this section.

Section	*Setup*	ΔR	*Rank sums*		
7.1	**Without LS**	17.51	***mmasResetT***	*mmasTreset* (65)	*mmasTuned* (178)
	With LS	27.65	***mmasTunedLS***	***mmasResetTLS*** (7)	*mmasTresetLS* (122)
7.2	**Without LS**	29.62	***mmasResetT***	*mmasMultiT* (98)	*mmasRestartT* (115)
	With LS	31.51	***mmasResetTLS***	*mmasMultiTLS* (34)	*mmasRestartTLS* (107)
7.3	**Without LS**	35.69	***mmasMultiRestartT***	***mmasMultiResetT*** (10)	*mmasMultiT* (56)
7.4	**Without LS**	36.31	***pacoTuned***	***mmasResetT*** (28)	*mmasMultiRestartT* (47)

value adopted by *mmasRestartT* is probably an effect of its strong forgetting behavior. Conversely, the very high ρ value adopted by *mmasMultiT* is likely induced from not having an explicit transfer mechanism.

Frequency of change and instance characteristics. The performance benefits provided by the different pheromone transfer approaches assessed in this section vary considerably as a function of the dynamic and structural characteristics of scenarios and instances. In general, the restart approach leads to less benefits than reset regardless of set or scenario, as illustrated in Figure 7 (top). By contrast, the multi-caste approach leads to the most significant improvements only on RUE instances when $f = 2$, as shown on Figure 7 (top left). In more detail, when we consider TSPLIB instances (for all scenarios) or even RUE instances for $f = 10$ scenarios, adopting multiple castes is the approach that brings less performance benefits among all transfer approaches compared. This is corroborated by the analysis provided in Table 5, where *mmasResetT* is the best-performing variant. Regarding the frequency of change, the reduced benefits from multiple castes when $f = 10$ are likely related to the split computational resources among ant groups, given that the runtime available for each slot is much smaller than when $f = 2$. Concerning instance characteristics, the experiments in Section 6 had already indicated that $\mathcal{MM}$AS needs more runtime for reoptimizing TSPLIB instances than for RUE instances, which becomes an issue for the multi-caste variant.

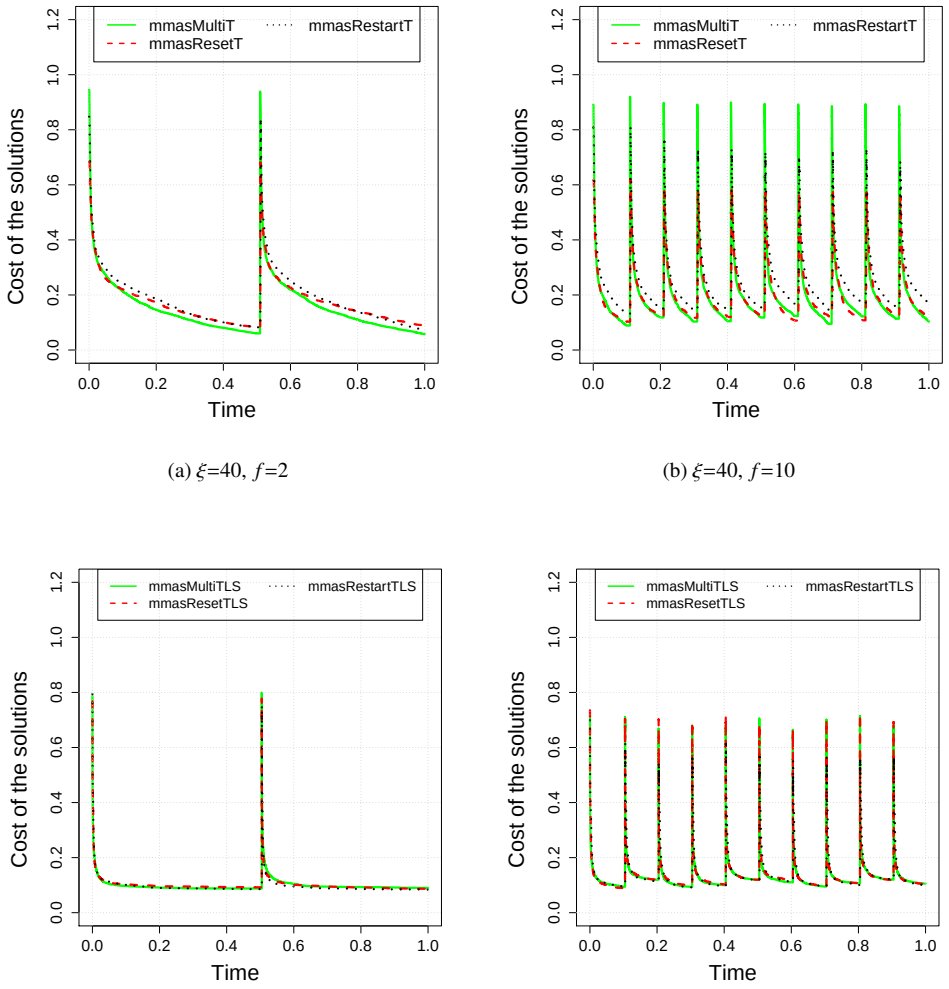
Local search. As in the preliminary assessment of pheromone transfer though reset, the presence of local search renders the benefits from the transfer approaches minimal. This is illustrated in Figure 7 (bottom), where we observe that this performance similarity between approaches holds whatever the frequency of change. Moreover, it is not possible to observe differences between transfer approaches whatever the instance set and degree of dynamism. Yet, when we aggregate over all runs considered, these small differences accumulate once again in favor of *mmasResetT*.

7.3. Combining pheromone transfer approaches

The assessment of pheromone transfer approaches produced two major insights. First, selecting a single pheromone transfer approach depends on the experimental factors adopted. Second, in the presence of local search, the benefits from all approaches considered become minimal. In this section, we assess the combination of pheromone transfer approaches, to see if it is possible to combine their advantages into a single variant. In more detail, the multi-caste approach only affects the solution construction rule. This way, the pheromone update behavior of the variants investigated in this section during a given environment is identical to the original $\mathcal{MM}$AS. Only when transitioning between environments is that the variants choose to either partially (*mmasMultiResetT*) or entirely (*mmasMultiRestartT*) forget the pheromone information from previous environments. Furthermore, we remark that we restrict our investigation in this section to experiments without local search.

Table 4 (bottom rows) shows the configured settings for the variants we consider in these experiments. The first variant is *mmasMultiT*, preliminarily assessed in the previous section, which we use as baseline. The second and third variants follow the multi-caste approach from the first variant, but explicitly promote pheromone transfer through reset (*mmasMultiResetT*) or restart (*mmasMultiRestartT*). Overall, all three variants present similar configuration, the strongest exception being parameter β, which is significantly increased for the novel variants. In a sense, $\mathcal{MM}$AS is now allowed to be more greedy as the transfer mechanisms help balance its search.

Figure 8 depicts SQT plots for experiments without local search, run on RUE instance 3002 on dynamic scenarios that present frequency of change $f = 10$, but differ as to the degree of dynamism (left: $\xi = 20\%$; right: $\xi = 80\%$). The most evident observation is how the combination of transfer approaches improve over using multiple castes alone. Two other observations stand out. First, it is rather interesting how the degree of dynamism affects the variants that



(a) ξ=40, f=2

(b) ξ=40, f=10

(c) ξ=40, f=2

(d) ξ=40, f=10

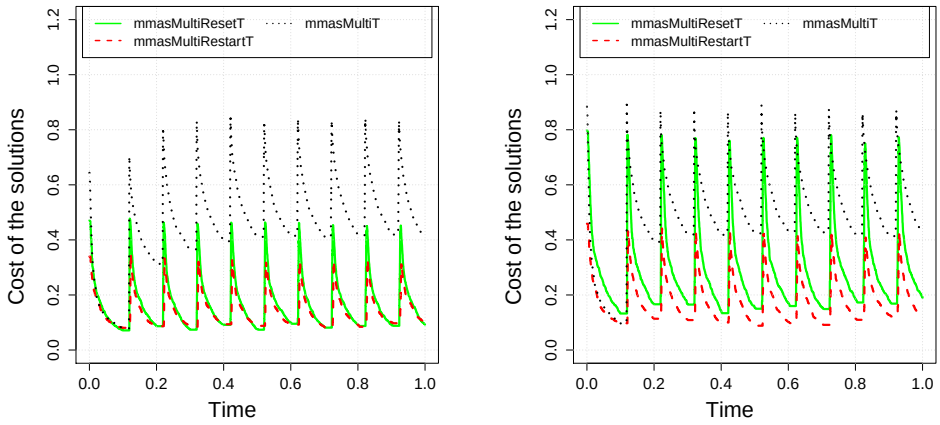
Figure 7: SQT plot depicting the anytime performance of $\mathcal{MM}$AS with three different versions of pheromone update mechanisms, run on TSPLIB instance pr2392 without local search, using same parameter configurations and different dynamic scenarios.

do not adopt restart. In the case of *mmasMultiT*, its reoptimization in the initial environments is more effective when the degree of dynamism is lower. In the case of *mmasMultiResetT*, this loss in performance is observed across all environments, with a competitive performance on $\xi = 20\%$ scenarios contrasting with a less competitive performance on $\xi = 80\%$ scenarios. Second, *mmasMultiRestartT* is able to improve over *mmasMultiResetT*, specially for larger RUE instance sizes. Altogether, these findings are an excellent evidence of algorithmic component interaction, given that in the previous set of experiments *mmasResetT* had outperformed both restart and multi-caste for a significant number of experimental factors.

‘

7.4. Comparing $\mathcal{MM}$AS variants with P-ACO

We conclude our investigation with a comparison of the best-performing algorithms and variants identified in this work. Namely, in this section we compare (i) P-ACO (*pacoTuned*); (ii) $\mathcal{MM}$AS coupled with pheromone transfer



(a) ξ=20, f=10

(b) ξ=80, f=10

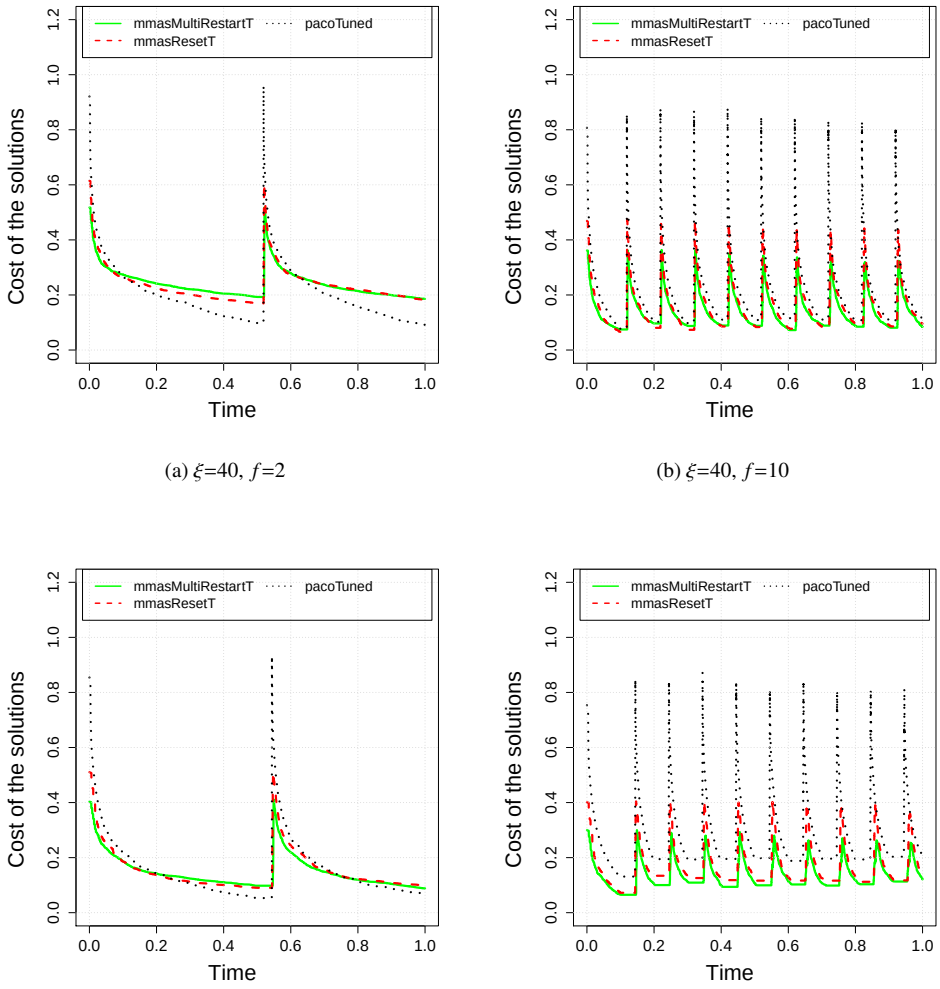
Figure 8: SQT plots depicting the anytime performance of three different $\mathcal{MM}$AS variants, run on RUE instance 3002 using different parameter configurations on $f = 10$ scenarios, with $\xi = 20\%$ (left) and $\xi = 80\%$ (right) .

through reset (*mmasResetT*), and; (iii) $\mathcal{MM}$AS using the multi-settings approach coupled with pheromone transfer through restart (*mmasMultiRestartT*). Note that these experiments do not consider local search, as none of the algorithms and variants considered here were competitive with (P-ACO) or able to improve significantly over (*mmasResetT* and *mmasMultiRestartT*) the original $\mathcal{MM}$AS coupled with local search. Our main objective here is then to measure the relative performance of the $\mathcal{MM}$AS variants w.r.t. P-ACO when local search is not adopted.

Figure 9 shows SQT plots depicting runs without local search on RUE instances 3002 (top) and 4002 (bottom), on scenarios that present degree of dynamism $\xi = 40\%$, but vary as to frequency of change (left: $f = 2$; right: $f = 10$). These plots help illustrate the two more relevant factors that affect results, as follows. First, the relative performance of the $\mathcal{MM}$AS variants w.r.t. P-ACO is strongly affected by the frequency of change. Specifically, Figure 9 (left) shows that P-ACO outperforms all $\mathcal{MM}$AS variants when $f = 2$. Conversely, we see from Figure 9 (right) that the $\mathcal{MM}$AS variants outperform P-ACO when there is an increase in the frequency of change. This is a rather remarkable result that confirms that $\mathcal{MM}$AS can be competitive in the context of dynamic optimization even in the absence of local search procedures.

The second factor that affect results is the number of cities. In particular, we observe that the instance size plays a more relevant role than the instance structural characteristics in this particular assessment. For this reason, we include both a smaller (top) and a larger (bottom) RUE instances, with the smaller instance being representative of the results for TSPLIB. When $f = 2$, P-ACO improves over the $\mathcal{MM}$AS variants by a larger margin on smaller instances (Figure 9a) than on larger instances (Figure 9c). Conversely, when $f = 10$ P-ACO is only competitive on smaller instances (Figure 9b), and is worse than the $\mathcal{MM}$AS variants by a significant gap on larger instances (Figure 9d) . When we look at the overall picture given in Table 5, we see that *pacoTuned* and *mmasResetT* are considered equivalent.

Altogether, these experimental factors suggest that the results from this comparison may have been influenced by the configuration setup adopted. More precisely, we have decided to use a single configuration for multiple experimental scenarios and benchmark instance sets. In addition, the TSPLIB instances we selected are smaller than the RUE instances we created. Balancing instance sizes among benchmark sets and adopting configurations specific to each scenario and set would likely help further understand the performances of these algorithms and variants. However, we have made a significant progress through this performance assessment, in that many factors that have been previously overlooked led to important insights, such as the effects of parameter configuration and the interactions between algorithmic components, such as pheromone transfer and local search.



(a) ξ=40, f=2

(b) ξ=40, f=10

(c) ξ=40, f=2

(d) ξ=40, f=10

Figure 9: SQT plot depicting the anytime performance of best performing $\mathcal{MM}$AS versions and P-ACO, run on RUE instance 3005 (9a, 9b) and on TSPLIB instance pr2392 (9c, 9d) without local search, using different parameter configurations and different dynamic scenarios. Left: scenario $\xi 40 f 2$. Right: scenario $\xi 40 f 10$.

8. Conclusions

The good performance of ant colony optimization (ACO) algorithms for optimization problems where the input data do not change over time motivated proposals to extend ACO to dynamic optimization. Among the most relevant algorithms proposed are those that re-evaluate the pheromone information deposited on the solution components affected by problem changes. The best known of these algorithms is P-ACO [22], which is based on a transient pheromone memory that quickly adapts exploiting the most recent solutions found. Another major share of the ACO dynamic optimization literature focuses on extending $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS [37]), given its excellent performance on static optimization. However, most existing works have overlooked important experimental factors, such as the proper configuration of parameter settings and the adoption of local search procedures, both critical to ACO performance in the context of static problems.

In this work, we have assessed P-ACO and many such *pheromone transfer* approaches through a rigorous computational study on a dynamic variant of the traveling salesman problem (TSP). Our work has produced, at least, three major contributions. First, we have proposed an automatic configuration approach for dynamic optimization, where we use the hypervolume as a metric to evaluate the anytime behavior of algorithms. Second, we have compared P-ACO to $\mathcal{MM}$AS, one of the best-performing ACO algorithms for the static TSP, isolating the effects of parameter settings and local search. In the absence of local search, as traditionally observed in the dynamic optimization literature, P-ACO outperforms $\mathcal{MM}$AS by a large margin. Yet, when local search is adopted, $\mathcal{MM}$AS consistently outperforms P-ACO on all experimental scenarios considered. Third, we have investigated the benefits of ACO adaptations for dynamic optimization, observing that their benefit is considerably reduced in the presence of local search. However, pheromone transfer approaches render $\mathcal{MM}$AS more effective than P-ACO for a large part of the experimental scenarios where local search is not adopted.

The insights produced in this investigation open a number of possible future work directions. Concerning the automatic configuration of anytime behavior for dynamic optimization, results revealed the importance of both instance characteristics and size, indicating better results could likely be obtained from isolating these factors for configuration. More importantly, given the overhead required by offline configuration, it would be paramount to assemble a repository of configured settings for multiple algorithms, benchmark sets, and experimental scenarios. In addition, we have only just started taking advantage of the theoretical benefits of the hypervolume. For instance, it remains an open question whether the Pareto-compliant properties of the hypervolume when measuring the anytime behavior within a single environment could also be extended for definite conclusions about the whole run.

Regarding the comparison between $\mathcal{MM}$AS and P-ACO, we have observed that conclusions are strongly dependent on local search. Yet, we have considered a single local search operator, which is neither the most efficient nor the most effective for the static TSP. As extended investigation on operators possibly better suited for dynamic optimization would likely indicate a means to reduce overhead while preserving the performance benefits. Conversely, in the absence of local search it was not possible to identify a single best-performing algorithm for all scenarios. Nonetheless, our work has paved the way for other high-performing ACO algorithms to be extended to dynamic optimization.

Acknowledgements

Authors Sabrina Moreira de Oliveira, Elizabeth Fialho Wanner, and Sérgio Ricardo de Souza would like to thank the Coordination for the Improvement of Higher Education Personnel (CAPES), the National Council of Technological and Scientific Development (CNPq), and the Federal Center of Technological Education of Minas Gerais (CEFET-MG) for supporting the development of the present study. Authors Marco Dorigo and Thomas Stützle acknowledge support from the F.R.S.-FNRS, of which they are research directors. This study was financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES) - Brazil - Finance Code 001.

Compliance and Ethical Standards

Authors Sabrina Moreira de Oliveira, Leonardo C. T. Bezerra, Marco Dorigo, Thomas Stützle, Elizabeth Fialho Wanner, and Sérgio Ricardo de Souza declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.

[2] M. Dorigo, M. A. Montes de Oca, S. Oliveira, T. Stützle, Ant colony optimization, in: J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, J. C. Smith (Eds.), Wiley Encyclopedia of Operations Research and Management Science, Vol. 1, John Wiley & Sons, Inc., 2011, pp. 114–125.

[3] P. Rohlfshagen, X. Yao, Attributes of dynamic combinatorial optimisation, in: X. Li, M. Kirley, M. Zhang, D. Green, V. Ciesielski, H. Abbass, Z. Michalewicz, T. Hendtlass, K. Deb, K. C. Tan, J. Branke, Y. Shi (Eds.), Simulated Evolution and Learning: 7th International Conference, SEAL 2008, Melbourne, Australia, December 7-10, 2008. Proceedings, Springer Berlin Heidelberg, Melbourne, Australia, 2008, pp. 442–451.

[4] S. Yang, Y. Jiang, T. T. Nguyen, Metaheuristics for dynamic combinatorial optimization problems, IMA Journal of Management Mathematics 24 (4) (2013) 451–480. doi:10.1093/imaman/dps021.
[5] M. Mavrovouniotis, S. Yang, M. Van, C. Li, M. Polycarpou, Ant colony optimization algorithms for dynamic optimization: A case study of the dynamic travelling salesperson problem [research frontier], IEEE Computational Intelligence Magazine 15 (1) (2020) 52–63.
[6] M. Mavrovouniotis, C. Li, S. Yang, A survey of swarm intelligence for dynamic optimization: Algorithms and applications, Swarm and Evolutionary Computation 33 (2017) 1 – 17.
[7] M. Mavrovouniotis, S. Yang, Ant colony optimization for dynamic combinatorial optimization problems, Swarm Intelligence 1 (2018) 121–142.
[8] H. N. Psaraftis, M. Wen, C. A. Kontovas, Dynamic vehicle routing problems: Three decades and counting, Networks 67 (1) (2016) 3–31. doi:10.1002/net.21628.
[9] C. J. Eyckelhof, M. Snoek, Ant systems for a dynamic TSP: Ants caught in a traffic jam, in: M. Dorigo, G. D. Caro, M. Sampels (Eds.), Proceedings of the Third International Workshop on Ant Algorithms (ANTS 2002), Brussels, Belgium, September 12-14 2002, Vol. 2463 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2002, pp. 88–99. doi:10.1007/3-540-45724-0_8.
[10] A. Simões, E. Costa, Memory-based CHC algorithms for the dynamic traveling salesman problem, in: N. Krasnogor, P. L. Lanzi (Eds.), Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO 2011), Dublin, Ireland, July 12-16, 2011, ACM, Dublin, Ireland, 2011, pp. 1037–1044. doi:10.1145/2001576.2001717.
[11] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, Applied Soft Computing 13 (10) (2013) 4023–4037. doi:10.1016/j.asoc.2013.05.022.
[12] A. Siemiński, M. Kopel, Solving dynamic tsp by parallel and adaptive ant colony communities, Journal of Intelligent & Fuzzy Systems 37(6) (2019) 7607–7618.
[13] M. Guntsch, M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic TSP, in: E. J. W. Boers, et al. (Eds.), EvoWorkshops 2001, Vol. 2037 of LNCS, Springer Verlag, Berlin, Germany, 2001, pp. 213–222.
[14] M. Guntsch, M. Middendorf, H. Schmeck, An ant colony optimization approach to dynamic TSP, in: L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt (Eds.), Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation (GECCO'01), San Francisco, California, USA, July 07-11, 2001, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 860–867.
[15] M. Mavrovouniotis, S. Yang, Ant colony optimization with immigrants schemes in dynamic environments, in: R. Schaefer, C. Cotta, J. Kolodziej, G. Rudolph (Eds.), Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN XI), Kraków, Poland, September 11-15, 2010, Part II, Vol. 6239 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010, pp. 371–380. doi:10.1007/978-3-642-15871-1_38.
[16] M. Mavrovouniotis, S. Yang, A memetic ant colony optimization algorithm for the dynamic travelling salesman problem, Soft Comput. 15 (7) (2011) 1405–1425. doi:10.1007/s00500-010-0680-1.
[17] L. A. Melo, F. B. Pereira, E. Costa, Multi-caste ant colony algorithm for the dynamic traveling salesperson problem, in: Proceedings of the 11th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA 2013), Lausanne, Switzerland, April 4-6, 2013, Vol. 7824 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 179–188. doi:10.1007/978-3-642-37213-1_19.
[18] M. Mavrovouniotis, S. Yang, Ant colony optimization with self-adaptive evaporation rate in dynamic environments, in: Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE 2014), Orlando, FL, USA, December 9-12, 2014, 2014, pp. 47–54. doi:10.1109/CIDUE.2014.7007866.
[19] M. Mavrovouniotis, S. Yang, Elitism-based immigrants for ant colony optimization in dynamic environments: Adapting the replacement rate, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2014, Beijing, China, July 6-11, 2014, 2014, pp. 1752–1759. doi:10.1109/CEC.2014.6900482.
[20] M. Mavrovouniotis, S. Yang, X. Yao, Multi-colony ant algorithms for the dynamic travelling salesman problem, in: Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014 IEEE Symposium on, 2014, pp. 9–16. doi:10.1109/CIDUE.2014.7007861.
[21] M. Mavrovouniotis, F. M. Müller, S. Yang, An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15, ACM, 2015, pp. 49–56. doi:10.1145/2739480.2754651.
[22] M. Guntsch, M. Middendorf, Applying population based ACO to dynamic optimization problems, in: M. Dorigo, G. D. Caro, M. Sampels (Eds.), Proceedings of the Third International Workshop on Ant Algorithms (ANTS 2002), Brussels, Belgium, September 12-14 2002, Vol. 2463 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2002, pp. 111–122. doi:10.1007/3-540-45724-010.
[23] M. Guntsch, *Ant Algorithms in Stochastic and Multi-Criteria Environments*, Ph.D. thesis, Universität Fridericiana zu Karlsruhe (2004).
[24] C. Li, M. Yang, L. Kang, A new approach to solving dynamic traveling salesman problems, in: T.-D. Wang, X. Li, S.-H. Chen, X. Wang, H. Abbass, H. Iba, G.-L. Chen, X. Yao (Eds.), Proceedings of the 6th International Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2006), Hefei, China, October 15-18, 2006, Vol. 4247 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2006, pp. 236–243.
[25] M. Mavrovouniotis, F. M. Müller, S. Yang, Ant colony optimization with local search for dynamic traveling salesman problems, IEEE Transactions on Cybernetics 47 (7) (2017) 1743–1756. doi:10.1109/TCYB.2016.2556742.
[26] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. Montes de Oca, M. Birattari, M. Dorigo, Parameter adaptation in ant colony optimization, in: Y. Hamadi, E. Monfroy, F. Saubion (Eds.), Autonomous Search, Springer, Berlin, Heidelberg, 2012, pp. 191–215. doi:10.1007/978-3-642-21434-98.
[27] M. Mavrovouniotis, S. Yang, Adapting the pheromone evaporation rate in dynamic routing problems, in: A. I. Esparcia-Alcázar (Ed.), Proceedings of the 16th European Conference on the Applications of Evolutionary Computation (EvoApplications 2013), Vienna, Austria, April 3-5, 2013, Vol. 7835 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2013, pp. 606–615. doi:10.1007/978-3-642-37192-961.
[28] S. Yang, Memory-based immigrants for genetic algorithms in dynamic environments, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO 2005), ACM, New York, NY, USA, 2005, pp. 1115–1122. doi:10.1145/1068009.1068196.

[29] W. Rand, R. L. Riolo, Measurements for understanding the behavior of the genetic algorithm in dynamic environments: A case study using the shaky ladder hyperplane-defined functions, in: Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation (GECCO'05), Washington DC, USA, June 25-26, 2005, ACM, New York, NY, USA, 2005, pp. 32–38. doi:10.1145/1102256.1102263.

[30] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (2012) 1–24. doi:10.1016/j.swevo.2012.05.001.

[31] J. P. Schmitt, F. Baldo, R. S. Parpinelli, A max-min ant system with short-term memory applied to the dynamic and asymmetric traveling salesman problem, in: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), IEEE, 2018, pp. 1–6.

[32] M. Mavrovouniotis, I. S. Bonilha, F. M. Müller, G. Ellinas, M. Polycarpou, Effective aco-based memetic algorithms for symmetric and asymmetric dynamic changes, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 2567–2574.

[33] J. K. Kordestani, A. Rezvanian, M. R. Meybodi, New measures for comparing optimization algorithms on dynamic optimization problems, Natural Computing 18 (4) (2019) 705–720.

[34] S. Oliveira, E. F. Wanner, S. R. de Souza, L. C. T. Bezerra, T. Stützle, The hypervolume indicator as a performance measure in dynamic optimization, in: K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, P. Reed (Eds.), Proceedings of the 10th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2019), East Lansing, MI, USA, March 10-13, 2019, Vol. 11411 of Lecture Notes in Computer Science, Springer, Cham, 2019, pp. 319–331. doi:10.1007/978-3-030-12598-1.

[35] F. S. Gharehchopogh, I. Maleki, M. Farahmandian, New approach for solving dynamic traveling salesman problem with hybrid genetic algorithms and ant colony optimization, International Journal of Computer Applications 53 (1) (2012).

[36] A. Radulescu, M. López-Ibáñez, T. Stützle, Automatically improving the anytime behaviour of multiobjective evolutionary algorithms, in: R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, J. Shaw (Eds.), Proceedings of the 7th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2013), Sheffield, UK, March 19-22, 2013, Vol. 7811 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2013, pp. 825–840.

[37] T. Stützle, H. H. Hoos, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, Future Generation Computer Systems 16 (8) (2000) 889–914. doi:10.1016/S0167-739X(00)00043-1.

[38] M. Dorigo, Optimization, learning and natural algorithms *(in italian)*, Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy (1992).

[39] C. Cruz, J. R. González, D. A. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, Soft Computing 15 (7) (2011) 1427–1448. doi:10.1007/s00500-010-0681-0.

[40] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (2012) 1–24. doi:10.1016/j.swevo.2012.05.001.

[41] T. Stützle, H. Hoos, Improvements on the ant-system: Introducing the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, in: Proceedings of the Third International Conference on Artificial Neural Nets and Genetic Algorithms (ICANNGA 97), Springer Vienna, Norwich, U.K, 1998, pp. 245–249.

[42] M. Mavrovouniotis, S. Yang, Memory-based immigrants for ant colony optimization in changing environments, in: C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. I. Esparcia-Alcázar, J. J. Merelo, F. Neri, M. Preuss, H. Richter, J. Togelius, G. N. Yannakakis (Eds.), Proceedings of the European Conference on the Applications of Evolutionary Computation: Applications of Evolutionary Computation (EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC), Torino, Italy, April 27-29, 2011, Part I, Vol. 6624 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2011, pp. 324–333. doi:10.1007/978-3-642-20525-5_33.

[43] A. Prakasam, N. Savarimuthu, Novel local restart strategies with hyper-populated ant colonies for dynamic optimization problems, Neural Computing and Applications 31 (1) (2019) 63–76.

[44] S. Yang, Genetic algorithms with elitism-based immigrants for changing optimization problems, in: M. Giacobini (Ed.), Proceedings of the Workshops on Applications of Evolutionary Computation (EvoWorkshops 2007) EvoCoMnet,EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Valencia, Spain, April 11-13, 2007, Vol. 4448 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007, pp. 627–636. doi:10.1007/978-3-540-71805-5.

[45] S. Yang, Genetic algorithms with memory- and elitism-based immigrants in dynamic environments, Evolutionary Computation 16 (3) (2008) 385–416. doi:10.1162/evco.2008.16.3.385.

[46] Xin Yu, Ke Tang, Xin Yao, An immigrants scheme based on environmental information for genetic algorithms in changing environments, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence) – CEC 2008, June 1-6, 2008, Hong Kong, China, 2008, pp. 1141–1147. doi:10.1109/CEC.2008.4630940.

[47] M. Mavrovouniotis, S. Yang, Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2012), Brisbane, Australia, June 10-15, 2012, 2012, pp. 1–8. doi:10.1109/CEC.2012.6252885.

[48] G. Tao, Z. Michalewicz, Inver-over operator for the TSP, in: A. E. Eiben, T. Bäck, M. Schoenauer, H. Schwefel (Eds.), Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - (PPSN V), Amsterdam, The Netherlands, September 27-30, 1998, Vol. 1498 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1998, pp. 803–812. doi:10.1007/BFb0056922.

[49] Á. E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 124–141.

[50] F. Hutter, H. Hoos, K. Leyton-Brown, T. Stützle, ParamILS: an automatic algorithm configuration framework, Journal of Artificial Intelligence Research 36 (2009) 267–306.

[51] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: C. A. C. Coello (Ed.), Proc. of LION-5, Vol. 6683 of Lecture Notes in Computer Science, Springer, 2011, pp. 507–523.

[52] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez-Cáceres, T. Stützle, M. Birattari, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58. doi:10.1016/j.orp.2016.09.002.

[53] H. Ben-Romdhane, E. Alba, S. Krichen, Best practices in measuring algorithm performance for dynamic optimization problems, Soft Computing 17 (6) (2013) 1005–1017. doi:10.1007/s00500-013-0989-7.

[54] G. Reinelt, TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (2008).

[55] H. Hoos, T. Stützle, Stochastic Local Search: Foundations and Applications, Morgan Kaufmann Publishers, San Francisco, CA, 2004.
[56] J. Branke, E. Salihoğlu, Ş. Uyar, Towards an analysis of dynamic environments, in: H. Beyer, U. O'Reilly (Eds.), Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05), Washington DC, USA, June 25-29, 2005, ACM, New York, NY, USA, 2005, pp. 1433–1440. doi:10.1145/1068009.1068237.
[57] W. Feng, T. Brune, L. Chan, M. Chowdhury, C. K. Kuek, Y. Li, Benchmarks for testing evolutionary algorithms, in: Proceedings of the 3rd Asia-Pacific Conference on Control and Measurement, Dunhuang, China, 31 Aug - 4 Sep 1998, 1998, pp. 134–138.
[58] K. Weicker, Performance measures for dynamic environments, in: J. J. Merelo Guervós, P. Adamidis, H. Beyer, J. L. F. Martín, H. Schwefel (Eds.), Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII), Granada, Spain, September 7-11, 2002, Vol. 2439 of Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2002, pp. 64–76. doi:10.1007/3-540-45712-7_7.
[59] R. W. Morrison, Performance measurement in dynamic environments, in: J. Branke (Ed.), Proceedings of the Workshop on Evolutionary Algorithms for Dynamic Optimization Problems (EvoDOP-2003) held in conjunction with the Genetic and Evolutionary Computation Conference (GECCO-2003), 12 July 2003, Chicago, USA, 2003, pp. 5–8.
[60] N. Mori, H. Kita, Y. Nishikawa, Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm, Transactions of the Institute of Systems, Control and Information Engineers 14 (1) (2001) 33–41. doi:10.5687/iscie.14.33.
[61] T. T. Nguyen, X. Yao, Continuous dynamic constrained optimization – The Challenges, IEEE Trans. Evolutionary Computation 16 (6) (2012) 769–786. doi:10.1109/TEVC.2011.2180533.
[62] E. Alba, B. Sarasola, ABC, a new performance tool for algorithms solving dynamic optimization problems, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2010), Barcelona, Spain, 18-23 July 2010, 2010, pp. 1–7. doi:10.1109/CEC.2010.5586406.
[63] J. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, TIK-Report 214, TIK-ETH Zürich, revised version (Feb. 2006).
[64] L. C. T. Bezerra, A component-wise approach to multi-objective evolutionary algorithms: from flexible frameworks to automatic design, Ph.D. thesis, IRIDIA, CoDE, Université Libre de Bruxelles (2016).
[65] B. Sarasola, E. Alba, Quantitative performance measures for dynamic optimization problems, in: E. Alba, A. Nakib, P. Siarry (Eds.), Metaheuristics for Dynamic Optimization, Vol. 433 of Studies in Computational Intelligence, Springer, Berlin, Heidelberg, 2013, pp. 17–33. doi:10.1007/978-3-642-30665-5_2.
[66] S. M. Oliveira, M. S. Hussin, T. Stützle, A. Roli, M. Dorigo, A detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP, in: GECCO (Companion)'11, 2011.
[67] S. Oliveira, M. S. Hussin, A. Roli, M. Dorigo, T. Stützle, Analysis of the population-based ant colony optimization algorithm for the tsp and the qap, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1734–1741.
[68] M. Dorigo, L. M. Gambardella, Ant colonies for the traveling salesman problem, BioSystems 43 (1997) 73–81.
[69] W. J. Conover, W. J. Conover, Practical nonparametric statistics, Wiley New York, 1980.