Information Sciences 180 (2010) 2390–2404

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An analysis of communication policies for homogeneous multi-colony ACO algorithms

C. Twomey, T. Stützle *, M. Dorigo, M. Manfrin, M. Birattari [1]

Université Libre de Bruxelles (ULB), IRIDIA, CP 194/6, Av. F. Roosevelt 50, B-1050 Brussels, Belgium

ARTICLE INFO

Article history:
Received 23 April 2009
Received in revised form 13 January 2010
Accepted 11 February 2010

Keywords:
Ant colony optimization
Parallelization
Traveling salesman problem
Communication topologies

ABSTRACT

The increasing availability of parallel hardware encourages the design and adoption of parallel algorithms. In this article, we present a study in which we analyze the impact that different communication policies have on the solution quality reached by a parallel homogeneous multi-colony ACO algorithm for the traveling salesman problem. We empirically test different configurations of each algorithm on a distributed-memory parallel architecture, and analyze the results with a fixed-effects model of the analysis of variance. We consider several factors that influence the performance of a multi-colony ACO algorithm: the number of colonies, migration schedules, communication strategies on different interconnection topologies, and the use of local search. We show that the importance of the communication strategy employed decreases with increasing search effort and stronger local search, and that the relative effectiveness of one communication strategy versus another changes with the addition of local search.

1. Introduction

Ant colony optimization (ACO) [17] is a metaheuristic for combinatorial optimization problems inspired by the pheromone trail laying and following behavior of some species of ants [13]. In ACO, artificial ants are a set of stochastic procedures that incrementally construct candidate solutions using artificial pheromone and, if available, heuristic information. The artificial pheromone is a parametrized probabilistic model that is modified at computation time based on previously seen solutions [45]. ACO algorithms have been tested on a large number of academic and real-world problems, and have obtained world-class performance on many of them, such as for some variants of the vehicle routing [21], sequential ordering [20], open-shop scheduling [3], and protein-ligand docking [26] problems. See [18] for a recent survey on ACO.

The cooperation of multiple colonies is a common adaptation of ACO to a parallel environment (see Section 3 for a more detailed discussion). In this arrangement, several ant colonies run in parallel, sharing information to focus the search on promising regions of the search space. However, some degree of independent exploration must be preserved, so that the computational effort of the colonies does not become wastefully concentrated in one region of the search space. To balance these two objectives, the colonies cooperate with a given *communication policy* specifying the details of what kind of information to exchange, when to exchange it, and among which colonies. A main focus of previous studies on multi-colony cooperation has been what kind of information to share and what communication topology to use to define each colony's neighbors. For the information to be exchanged, the results in Krüger et al. [27] and Doerner et al. [14] indicate that it is

* Corresponding author.
E-mail addresses: ctwomey@princeton.edu (C. Twomey), stuetzle@ulb.ac.be (T. Stützle), mdorigo@ulb.ac.be (M. Dorigo), max.manfrin@gmail.com (M. Manfrin), mbiro@ulb.ac.be (M. Birattari).
[1] The ordering of the authors' names was chosen at random.

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.02.017

better to exchange the best solutions found than to exchange pheromone matrices. For the topologies to be used, the ring and the fully-connected topologies are among the most tested topologies in the literature.

Unfortunately, many previous studies on multi-colony cooperation suffer from a lack of rigorous empirical analysis. In some cases, empirical tests are done on only very few instances of small size, as in [8,11,43,44]. Other studies do not make comparisons with the non-cooperative policy, i.e., the parallel independent runs on multiple processors of the single colony algorithm (PIR), as in [41]. Thus, it is difficult to draw general conclusions from the results presented in the literature (again, see Section 3 for a more detailed discussion). This situation leaves open the question of how best to implement efficient parallel versions and what improvement in performance can be expected over PIR.

The most complete study on the cooperation of multiple colonies is that of Middendorf et al. [33,34], which investigated four communication policies for a homogeneous multi-colony approach based on a variant of ant system to solve the traveling salesman problem and the quadratic assignment problem. The study compared the multi-colony ant algorithm with PIR, with the result that, if a high solution quality is desired, "...information exchange between the colonies is useful but otherwise no information exchange is better" [34, p. 319], which we interpret to mean that increasing levels of communication for equivalent computational effort results in better solution quality than parallel independent runs of the same algorithm.

On the basis of our preliminary results [29,30], we believe that the conclusions obtained in [34] do not extend to cooperative multi-colony ACO algorithms composed of high-performing sequential implementations employing a local search component. Hence our hypothesis: *the cooperation of multiple homogeneous colonies becomes less effective for increasing search effort and stronger local search algorithms*. We present a detailed study that analyzes the impact various communication policies for parallel homogeneous multi-colony ACO algorithms have on solution quality. We extensively test the parallel variants on a distributed-memory parallel architecture, and we analyze the impact of the factors studied on the solution quality with a fixed-effects model of the analysis of variance (ANOVA). The four factors considered are: the number of colonies (2 levels), the migration schedule (2 levels), the communication strategy (4 levels), and the local search component (3 levels). All together, the factors result in $2 \cdot 2 \cdot 4 \cdot 3 = 48$ different algorithmic configurations. As the test problem we have chosen the traveling salesman problem because of its central role in ACO research.

The rest of the paper is organized as follows: in Section 2, we briefly describe the ACO framework and the $\mathcal{MAX}$–$\mathcal{MIN}$ ant system variant that we use for our parallel implementations. In Section 3, we review different communication policies proposed in the ACO literature. Section 4 presents the policies we investigated, and Section 5 describes the experimental setup. The empirical results of our parallel implementations are given in Section 7, while scalability tests for the most successful policies (with up to 64 colonies) are discussed in Section 8. In Section 9, we report the results of confirmatory experiments on the predictions made from our earlier results. In Section 10, we discuss the implications of the overall results.

2. ACO for the traveling salesman problem

The traveling salesman problem (TSP) is an $\mathcal{NP}$-hard combinatorial optimization problem [28] that has often been used as a benchmark to test new algorithmic ideas in ACO and other stochastic local search methods [24]. Formally, it is the problem of finding a Hamiltonian cycle of minimal length on a complete weighted graph $G = (V, E)$, where each vertex $v \in V$ of the graph represents a location and each edge $e \in E$ represents a connection between two locations. Each edge e is assigned a weight c_e, which represents the distance between the locations it connects.

ACO is an iterative procedure. In every iteration, each ant of the colony constructs a solution to the given problem. When applied to the TSP, each ant is initially placed on a randomly chosen vertex and has a memory to store the partial solution it has constructed so far in the iteration. Each ant performs a construction step by moving from its current vertex to one of its still unvisited vertices. At each construction step, an ant chooses stochastically among the unvisited vertices. Each edge has associated with it two kinds of information to bias the movement of the ants: artificial pheromone information (that can be modified by ants) and heuristic information (usually a measure inversely proportional to the distances between the locations). An ant $k \in \{1, \cdots, m\}$ moves from vertex i to vertex j with a probability given by:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in \mathcal{N}_k} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, & \text{if } j \in \mathcal{N}_k; \\ 0, & \text{otherwise;} \end{cases} \tag{1}$$

where $\mathcal{N}_k$ is the set of vertices not yet visited by the k-th ant, τ_{ij} is the value of the artificial pheromone information associated to the edge connecting vertex i to vertex j, and η_{ij} is the value of the heuristic information associated with the same edge. The parameters α and β determine the relative importance of the pheromone information and heuristic information.

At the end of an iteration, the pheromone on the edges is updated. The pheromone update consists of two parts. First, a pheromone evaporation is performed, which uniformly decreases all the pheromone values to avoid a too rapid convergence of the algorithm. Then, one or more solutions from the current and/or earlier iterations are used to increase the values of pheromone information on the edges of these solutions.

In $\mathcal{MAX}$–$\mathcal{MIN}$ ant system ($\mathcal{MM}$AS), only the best ant is used to update the pheromone trails, and the minimum and maximum values of the pheromone are limited. A design choice is whether to use the ant with the best tour found in the current iteration —*iteration-best* update—or the ant with the best tour found among all iterations so far—*best-so-far* update—or even the ant with the best tour found since the last re-initialization of the pheromone information—*restart-best*

update—or a combination of all of them. The pheromone trails are initialized to the maximum pheromone value to allow for greater exploration during the initial phase of the search—or when the pheromone matrix is re-initialized, which is occasionally done in $\mathcal{MM}$AS. The pheromone update for the edge (i,j) is performed as follows:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{\text{best}}, \tag{2}$$

where $0 < \rho \leqslant 1$ is the evaporation rate and $\Delta\tau_{ij}^{\text{best}}$ is the quantity of pheromone deposited on edge (i,j) by the best ant. Formally:

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} \frac{1}{L_{\text{best}}}, & \text{if the best ant used edge } (i,j) \text{ in its tour;} \\ 0, & \text{otherwise;} \end{cases} \tag{3}$$

where L_{best} is the length of its tour.

For what concerns the limits on the pheromone values, respectively, $\tau_{\min}$ for the minimum and $\tau_{\max}$ for the maximum, Stützle and Hoos [40] suggest that they should be chosen according to the problem instance at hand from among the general values they have determined.

The pheromone update process in $\mathcal{MM}$AS is concluded by enforcing that all pheromone values are within the imposed limits.

3. Related work on parallel ACO

The use of ACO or other stochastic local search methods to solve optimization problems may require significant computational effort. For this reason, parallel implementations of such algorithms are desirable. Within the ACO framework, several opportunities for parallelization exist. In ACO, each artificial ant builds a solution independently from other ants (with the exception of ant colony system, proposed by Dorigo and Gambardella [16]). Hence, executing the construction of the solutions in parallel seems to be a straightforward way to parallelize the algorithm. Most of the early studies on ACO parallelization focus on this possibility. The primary goal of such an approach is to take advantage of the increased computing resources to reduce the computation time of a single run of the algorithm. This approach has been tested on a variety of parallel architectures, with both shared [4,12,15] and distributed [4,36] memory. However, it was found that very often the communication overhead neutralizes the advantages of parallelizing the construction phase. Inspired by work in the evolutionary computation community (such as the work done on distributed genetic algorithms [2,42]), ACO researchers have begun to investigate a different approach: the cooperation of multiple colonies.

As described below, several researchers have investigated policies for sharing information in multi-colony ACO algorithms to find the best method of cooperation for achieving the highest overall performance. To define a policy one has to specify each of the following elements:

- *migrants*: what kind of information should be exchanged between colonies (e.g., the colony's parameters, solutions, or pheromone information);
- *policy for migrants selection*: how many and which migrants should be selected from the source colonies;
- *strategy*: which colonies should send migrants to which other colonies (e.g. global-best, replace-worst, uni/bi-directional ring, hypercube, grid, or random connection topologies);
- *migration schedule*: when migrants should be sent/received;
- *policy for migrant integration*: how many and which migrants should be integrated into the target colonies.

The simplest policy is the null policy, i.e., the parallel independent execution on multiple processors of the single colony algorithm. This approach—parallel independent runs (`PIR`)—is appealing because no communication overhead is involved and nearly no additional implementation effort is necessary. In [38], Stützle has shown that such a simple policy can be highly efficient in the case of a $\mathcal{MM}$AS algorithm for solving the TSP. More elaborate mechanisms are justified if they give better performance than the execution of parallel independent runs. Therefore, a comparison with the `PIR` policy should always be performed when introducing a new communication policy.

Michel and Middendorf's study [32] is one of the first investigations on information sharing among multiple ant colonies. The authors implemented a variant of ant system (AS) with elitist strategy to solve the shortest common supersequence problem. In their approach, each colony selects for migration its best-so-far solution. Following a global-best strategy (each colony is the neighbor of all the other colonies and the overall best-so-far solution is distributed in the neighborhood), migrants are circulated with a high fixed-frequency migration schedule. The policy for migrants integration is such that a received migrant replaces the local best-so-far solution. A small difference in solution quality from the single colony approach (`SEQ`) was found, but no statistical analysis was provided to assess the significance of the difference.

In subsequent studies, Middendorf et al. [33,34] further investigated communication policies for a multi-colony approach based on a variant of AS for solving the TSP and the quadratic assignment problem. They chose the colony best-so-far (and/or a certain number of elitist) solutions to be their migrants, a fixed-frequency migration schedule, and two different strategies for comparison: global-best and unidirectional ring. The policy providing the better results was compared with a `PIR` approach, but was empirically validated on only a single instance of the TSP—a clear shortcoming in their empirical analysis. The overall best policy found was the unidirectional ring strategy with exchange of the colony best-so-far solution, with the

conclusion that, for the TSP, no information exchange is an advantage for quickly finding solutions of lower quality, while information exchange is advantageous for finding solutions of high quality.

Other studies investigating multi-colony approaches using fixed-frequency migration schedules have found mixed results with regards to the comparison of the communication policies considered and PIR and/or SEQ. Such experiments have been conducted by, among others, Piriyakumar and Levi [35] and Manfrin et al. [29], using a multi-colony variant of $\mathcal{MM}$AS with local search. In both studies, little difference was found between the considered policies and SEQ, and even some indication in Manfrin et al. that PIR was better than cooperation. In contrast, Chu et al. [8,9] and Ellabib et al. [19], both using ant colony systems (ACS) for the TSP (and the vehicle routing problem in [19]), found that in fact some communication policies perform significantly better than SEQ.

The difference in result may in part be due to the strength of the multi-colony variant of $\mathcal{MM}$AS when coupled with local search relative to that of ACS for the TSP, but this is not clear from the literature. Both Piriyakumar and Levi and Chu et al. share a significant shortcoming in their empirical analysis: they only use a very small set of TSP instances (one and three, respectively) for their trials. In the latter case it must also be noted that the parameter settings used for the single colony ACS are known to result in poor performance, and, hence, the results presented are not very conclusive.

While it may be unclear what the impact an ACO variant's strength has on the best communication policy to be used, it also remains unclear what frequency is best to use for a fixed-frequency migration schedule, and whether or not a fixed-frequency migration schedule is even the appropriate choice. Doerner et al. [14] investigated different frequencies of information exchange using the fixed-frequency policy for a multi-colony implementation based on the savings-based AS, and a parallel version of D-Ants (an approach that decomposes the original problem into smaller sub-problems, which are then solved in parallel). From their study, it appears that for the fastest schedule the exchange of the overall best-so-far solution led to the best results, while for the slowest schedule there was a dependency on the number of colonies: when two colonies were used, the exchange of the overall best-so-far solution and elitist solutions led to the best results; when four colonies were used, the re-initialization of the pheromone matrices before the exchange of the solutions led to the best results.

Rather than use a fixed-frequency schedule, Chen and Zhang [7] proposed two communication policies for a multi-colony approach, based on a variant of AS for solving the TSP, employing a variable-frequency migration schedule and two interconnection topologies. The proposed migration schedule adjusts the time interval between two exchanges adaptively according to the diversity of the solutions. The diversity in a colony is measured as the ratio between the average colony's solution cost and the colony's best-so-far solution cost. The diversity for the whole system is the average diversity among the colonies. The less (more) diversity in the whole system, the shorter (longer) the time interval for the next exchange. The policies were compared with: the standard sequential ant system, the unidirectional ring topology, the exchange of the colony best-so-far solutions of [34], and a fixed-frequency migration schedule for different intervals. The results indicate that the adaptive variable-frequency schedule performs better than the others considered, but it must be noted that the comparisons were conducted for only five TSP instances of small size, which weakens the generality of their results.

As pointed out in Section 1, some of the previous studies presented in the literature suffer from a lack of rigorous empirical analysis. Tests were conducted on only a few instances of often small size, there was a lack of comparisons to control experiments (e.g. with PIR), or tests were conducted with parameter settings that are known to result in poor performance. It is therefore difficult to compare and generalize from these results.

Consequently, the question remains open as to how best to design efficient communication policies for parallel versions of ant colony optimization, especially with regards to the inclusion of stronger local search. This study will address this question with a more comprehensive approach than what has been attempted before, using a significantly larger range of problem instances, a broad set of communication policies, both fixed and variable frequency migration schedules, a very high performing ACO algorithm—$\mathcal{MM}$AS, and the incorporation of local search. These factors will be carefully controlled for in an experimental study with parallel independent runs serving as a meaningful baseline for comparison.

4. Considered communication policies

In our investigation, we fixed the following elements in the considered communication policies: each colony selects as the only *migrant* its best-so-far solution, and the *policy for migration integration* is such that a received solution becomes the new colony's best-so-far solution (and the new colony's restart-best solution) if and only if the received solution has a lower cost than the colony's current best-so-far solution.

For the other elements, in order to limit the explosion in the number of possible policies, we restricted our investigation to two levels for the factors *number of colonies* and *migration schedules*, and to four levels for the element *communication strategies*.

We examined a number of *interconnection topologies* that allowed us to consider increasing communication volumes. The considered topologies were the *directional ring*, in which each colony has one predecessor and one successor, the *hypercube*, in which each colony is located on a vertex of a hypercube (see [22] for a detailed explanation of the topology) and communicates only with colonies that are located in adjacent vertices, and the *fully-connected* topology, in which each colony is in the neighborhood of every other colony. In this way, we moved from more localized to more global communication. On the three considered topologies (Fig. 1) we implemented the following four *communication strategies*:

- *Unidirectional ring* (R). The p colonies are connected in a ring such that colony C_i sends its best-so-far solution only to colony $C_{(i+1) \bmod p}$, and receives a best-so-far solution only from colony $C_{(i-1+p) \bmod p}$.

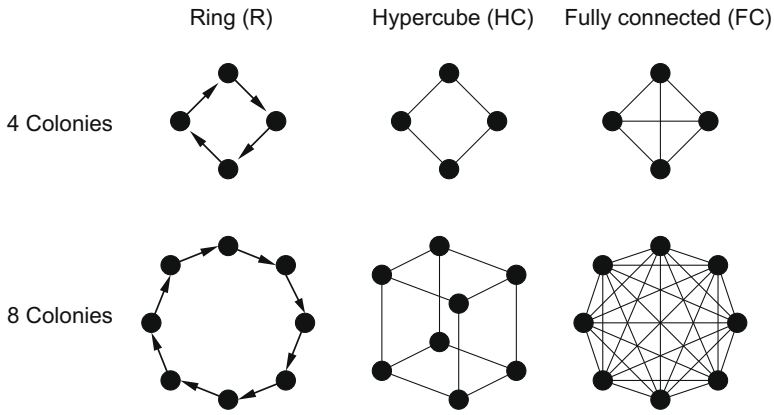

Fig. 1. The communication topologies under consideration: unidirectional ring (R), hypercube (HC), and fully-connected (FC), for each of the two different levels of *number of colonies*: 4 and 8. (Replace-worst (RW) is implemented on the FC topology, with the difference that only at most one node is changed each iteration.)

- *Hypercube* (HC). Each colony is located on a vertex of a hypercube (2-D hypercube for $p = 4$ and 3-D hypercube for $p = 8$) and sends its best-so-far solution to the colonies that are located in adjacent vertices. It receives a best-so-far solution from each colony located in adjacent vertices.
- *Replace-worst* (RW). One colony acts as a *master* collecting the cost of the best-so-far solutions of the other $p - 1$ colonies. The *master* identifies and broadcasts the identity of the colony with the overall best-so-far solution and the colony with the worst best-so-far solution. At this point, the former colony sends its best-so-far solution to the latter colony.
- *Fully-connected* (FC). This parallel model is similar to RW, with the difference that the *master* identifies and broadcasts the identity of the colony with the overall best-so-far solution, that subsequently broadcasts its best-so-far solution.

These communication topologies can be found also in the multi-population based evoluationary algorithms (EAs) or the well known island model for parallelizing evolutionary algorithms [5,1]. We chose these particular topologies for the same reasons that they are prominent in the EA literature: they are representative of increasing levels of communication, as defined by their outdegree and diameters. The ring, hypercube, and fully-connected topologies have increasing outdegrees $(1, \log_2(N)$, and $N - 1$, respectively, where N is the number of vertices), and decreasing diameters $(N, \log_2(N)$, and 1, respectively). Choosing topologies with properties other than increasing outdegree and decreasing diameter may be interesting as a future line of research, but is well beyond the scope of this work. Although we use topologies that are well studied in the EAs literature, the theoretical and empirical results found for EAs with regards to optimally selecting, for example, the population size based on communication topology (as in [6]) should not be expected to apply to ACO. Note that these communication topologies are also used as neighborhood topologies in Particle Swarm Optimization algorithms [25]. However, there the usage is very different from the one in parallel ACO algorithms since, in PSO, communication takes place after each iteration of the algorithm and communication in non-parallel PSO is among individual solutions and not among sets of solutions.

For the *number of colonies* we considered two levels: 4 and 8 identical colonies. In Section 8, we report on the tests to check if performance trends identified using these two levels were maintained for the most successful policies when the *number of colonies* was increased to 16, 32 and 64.

The two levels used for the *migration schedules* were (1) a fixed-frequency schedule and (2) an increasing-frequency schedule. The rationale for an increasing-frequency schedule is to have colonies focusing on promising areas of the search space during the later intensification phase of the search. Colonies perform the initial T iterations without exchanging any solution, to favor an independent exploration of the search space. In the fixed schedule, colonies exchange solutions every $c > 0$ iterations. In the increasing schedule, solutions are exchanged with increasing frequency, but taking care that at least c iterations pass between two exchanges, where $0 < c < T$. The ith exchange happens at iteration $\sum_{k=0}^{i-1} \lfloor b^k \cdot T \rfloor_c$, with $0 < b \leqslant 1$, and where $\lfloor x \rfloor_c = c$ if $x < c, \lfloor x \rfloor_c = \lfloor x \rfloor$ otherwise, where $\lfloor x \rfloor$ is the largest integer not exceeding x. For example, when $T = 1000, b = 0.9$, and $c = 25$ the first 10 exchanges happen at iterations 1000, 1900, 2710, 3439, 4095, 4685, 5216, 5694, 6124, 6511. Note that the minimum bound of 25 iterations between communications is obtained from iteration 9735 on.

5. Experimental setup

For our investigation, we considered $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, one of the best-performing sequential variants of the ACO framework for the TSP. Our parallel implementation of $\mathcal{MM}$AS is based on the publicly available ACOTSP software.[2] We extended the ACOTSP code by adding a quadrant nearest neighbor candidate list of size 20, and we modified the pheromone

[2] The source code for the current version of ACOTSP is licensed under the GNU General Public License and is available for download at http://www.aco-metaheuristic.org/aco-code/public-software.html.

Table 1
Full-factorial design: configurations are obtained considering all combinations of factor levels.

Number of colonies	Communication strategy	Migration schedule	Local search
4	R	Fixed	none
8	HC	Increasing	2-opt
	RW		3-opt
	FC		

update schedule in order to schedule only best-so-far and restart-best solutions.[3] The parameters of $\mathcal{MM}$AS were chosen in order to guarantee robust performance over different instance sizes, and follow the ones proposed in [40] for the case with local search: $\alpha = 1, \beta = 2, \rho = 0.2$,[4] and $m = 25$.[5] The two considered schedules for migration frequency were the fixed schedule with $T = 100$ and $c = 25$, and the increasing schedule with $T = 1000, b = 0.9$, and $c = 25$.

Each algorithm was tested with three levels of the *local search* component: `none`, `2-opt` and `3-opt`. The *local search* component was applied to all the solutions constructed in each iteration. Experiments were performed on a homogeneous cluster with 4 units, each featuring two AMD Opteron™ 244 CPUs with 2 GB of RAM. The processing nodes were connected by a 1 Gbit Ethernet communication network that allowed every processing node to communicate directly with every other processing node. Each unit was running the Rocks Cluster Linux 4.2.1 operating system at the time of the experiments.[6]

Properties of the problem instances to be solved can have an impact on the performance of the algorithm. From the benchmark library TSPLIB [37], we considered various instances with different properties (the specific names are provided in Sections 6 and 7). The optimal solution is known for all the chosen instances. For each instance, 30 runs were performed in order to create a large enough sample size to allow for a statistical analysis of the results. The results report percentage error from the optimal cost.

For experiments involving metaheuristics, it is a common practice to have a specific bound for the computational effort (a maximum CPU time or a maximum number of iterations). When not indicated differently, our algorithms were stopped after each colony performed 10,000 iterations. To evaluate the quality of a communication policy, the *parallel independent runs* (`PIR`) variant was used as a baseline for comparison. In `PIR`, 4 or 8 runs of the same single colony $\mathcal{MM}$AS algorithm are simultaneously and independently executed using different random seeds. The final result is the overall best-so-far solution.

The naming convention for the configurations is the following: a number, indicating how many identical ant colonies are used (`4` or `8`), followed by the communication strategy label of the algorithm (`R`, `HC`, `RW`, `FC`, and `PIR`), followed by a letter indicating the adopted migration schedule (`f` for fixed, `i` for increasing), followed, finally, by a number indicating the adopted local search (`0` for no local search, `2` for the `2-opt` local search, and `3` for the `3-opt` local search). The reference algorithm implementing the `PIR` strategy does not have the migration frequency schema's letter in its name because `PIR` has no migration schedule. According to this naming convention, the label `4RWf2` identifies a configuration with 4 identical colonies, a replace-worst strategy on a fully-connected topology, adoption of the fixed-frequency migration schedule, and `2-opt` local search; while the label `8PIR3` identifies an algorithm with eight identical colonies, a parallel independent runs strategy and `3-opt` local search. In Section 6, where `SEQ` was used, no colony number pre-fix is necessary, nor is a migration frequency identifier letter. Thus, the label `SEQ0` identifies a configuration with 1 colony that uses no local search. In Table 1, we show the factorial design of the experiments considered in this investigation. Adopting a full-factorial design, the number of considered configurations is $2 \cdot 4 \cdot 2 \cdot 3 = 48$.

6. Preliminary experiments

When dealing with the traveling salesman problem and many other $\mathcal{NP}$-hard combinatorial optimization problems, local search is known to play a major role in improving the performance of ACO algorithms. To assess the impact of this component on the final solution quality in our implementation of $\mathcal{MM}$AS, we compared `SEQ0`, `SEQ2`, and `SEQ3` on four TSPLIB instances. To conduct a fair comparison we used a stopping criterion based on maximum CPU time rather than a maximum number of iterations due to the different time complexities of the three algorithms. The average number of iterations completed in the considered time is listed in Table 2.

[3] Experiments performed by one of the authors indicate that for large instances the schedule without iteration-best solutions does not worsen the quality of the solution.

[4] Note that the usage of ρ in [40] refers to an equation $\tau_{ij} = \rho \cdot \tau_{ij} + \cdots$, while the ACOTSP package uses the form $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \cdots$. Hence, the recommended setting in [40] for $\mathcal{MM}$AS with local search corresponds to the setting of ρ we use in the ACOTSP implementation of $\mathcal{MM}$AS.

[5] The parameter setting does not follow the recommendations in [40] with respect to ρ and the number of ants in the case no local search is used. This choice was done in order to simplify the experimental setup and to avoid optimizing the parameter settings for different instance sizes. In fact, $\mathcal{MM}$AS parameters settings in the case without local search need to be adapted when tackling larger instances to force convergence of the algorithm within a practical computational budget [39] (see p. 75).

[6] The sources were compiled in the following environment: **glibc** 2.3.4, **gcc** 3.4.6, with optimization flag `-O3`, and the **LAM/MPI** 7.1.1 communication libraries.

Table 2
Instances and corresponding maximum CPU time (search time). Number of iterations done by each algorithm for each instance, averaged over 30 runs (iterations).

Instance	Search time (s)	Iterations		
		SEQ0	SEQ2	SEQ3
lin318	60	8577	5842	2035
rat783	300	12,957	9458	4522
nrw1379	1200	20,460	16,185	7277
pr2392	1800	17,912	16,977	7604

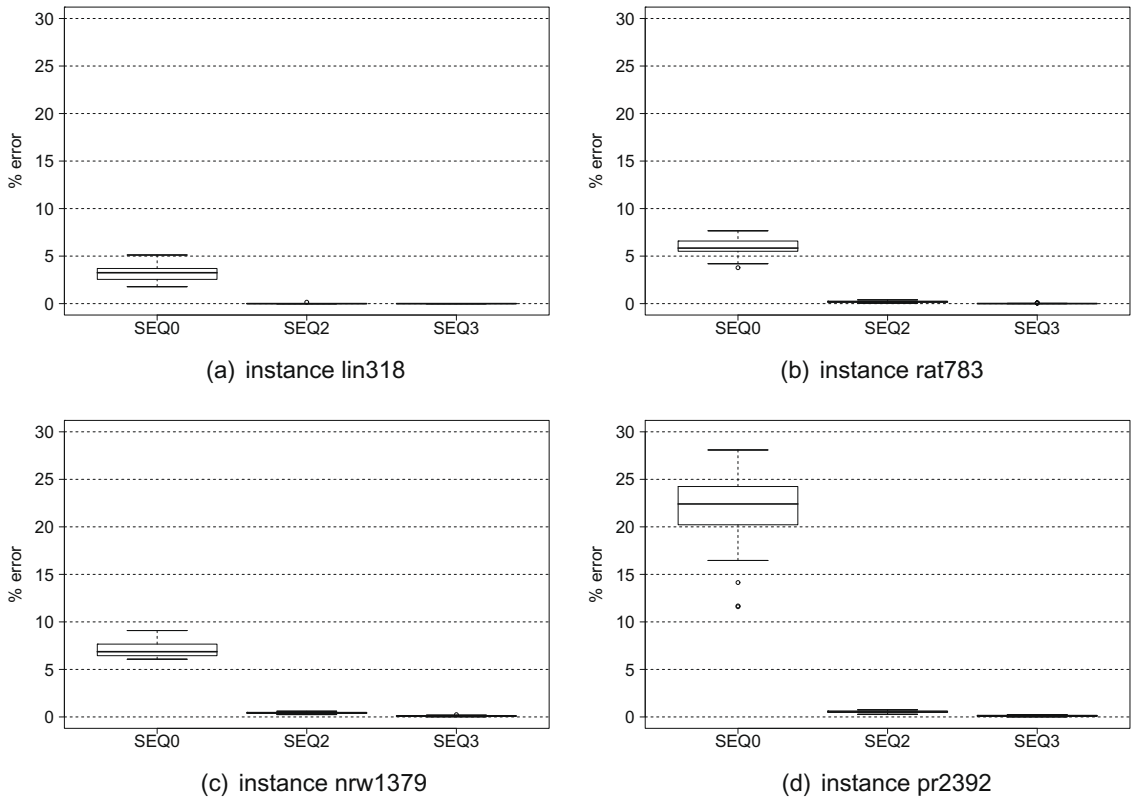

Fig. 2. Boxplot over 30 independent trials of the sequential $\mathcal{MM}$AS algorithms using: no local search (SEQ0), 2-opt local search (SEQ2), and 3-opt local search (SEQ3).

We used a *two-sided pairwise Wilcoxon rank sum test* [10] with *p-values* adjusted by Holm's method [23] to assess the statistical significance of the differences in solution costs obtained by the algorithms under analysis.

Fig. 2 contains the boxplot of normalized solution costs with respect to the percentage error from the known optimal cost. Full raw data and statistical analyses are available in [31]. The differences are statistically significant for all the instances with the exception of the results of SEQ2 and SEQ3 on the smallest instance (lin318). The boxplots confirm that SEQ3 achieves, on average, the best performance, while SEQ0 obtains the worst. We observe that the larger the instance, the wider the gap in performance between SEQ0 and SEQ2, SEQ3. It appears that the local search component and problem instance are factors with excessive effects and, therefore, to better comply with the ANOVA assumptions, we will analyze the configurations in separate groups according to the problem instance and to the levels of the local search factor.

7. Experimental results

As suggested in the previous section, we analyzed the results of the various configurations in separate groups according to the levels of the local search factor and the problem instance. In each of these analyses we compared the different configurations with PIR by means of *two-sided pairwise Wilcoxon rank sum tests* with *p-values* adjusted by Holm's method in order to evaluate the quality of the communication policies. To investigate the impact of the computational effort on the solution

cost obtained by each policy, we performed the comparison considering the results after 1000 iterations (short runs), after 3162 iterations (medium runs), and after 10,000 iterations (long runs). For those readers interested in the full raw data and statistical analyses, we refer to [31].

A common element in all analyses is that, for short runs, we did not find statistically significant differences between the configurations using an increasing-frequency migration schedule and PIR according to the *Wilcoxon test*. This was expected since for those configurations colonies do not exchange solutions during the first 1000 iterations, and thus behave like PIR.

A second element common to all analyses is that, as expected, the factor *number of colonies* produces consistently better results for a level of eight rather than four, and, therefore, we will not report on this result in the following.

Lastly, ANOVA analyses were conducted for each of the seven TSPLIB instances, at each level of local search, considering the results for short, medium and long runs, separately. The outcomes of the ANOVA analyses are summarized in Table 6, and are taken into account in the following presentation of the results.

7.1. Configurations without local search

In a first set of experiments, we compared the configurations that do not use local search. The experiments were carried out on seven TSPLIB instances: kroA100, eil101, kroA200, lin318, pcb442, att532, and rat783 within the experimental framework described in Section 5.

For short runs, the differences in performance between configurations using a fixed-frequency schedule and that of PIR are statistically significant according to the *Wilcoxon test* (Table 3), achieving better performance than PIR (with exceptions for some small instances). When the computational effort was extended to medium runs, the fixed-frequency schedule continued to achieve better performance than PIR, with the exception of the smallest instance kroA100, for which it performed worse. For long runs, configurations using the increasing-frequency schedule matched the fixed-frequency schedule configurations in terms of the number of statistically significantly better performances than PIR, while the fixed-frequency schedule configurations had more statistically significantly worse performances than PIR compared to the increasing-frequency schedule configurations. Overall, in the long run case there was a reduction in the number of configurations that performed better than PIR. When communication policies were found to have an advantage over PIR, it was typically on the largest instances considered. More generally, greater amounts of communication (here achieved with FC and HC, as opposed to R and RW) in multi-colony ACO algorithms without local search seems to have a positive impact on solution quality, supporting earlier work in the literature [34]. The magnitude of the impact appears to be mainly dependent on the computational effort and on the *migration schedule*. As can be seen from the values in Table 3, the beneficial effects of communication with a fixed-frequency migration schedule are overall bigger but tend to be reduced when the computational effort is increased. In contrast, when an increasing-frequency schedule is adopted, the beneficial effects tend to grow with increasing computational effort. The overall best *communication strategies*, on average, are the replace-worst and the unidirectional ring with the fixed-frequency migration schedule.

7.2. Configurations with 2-opt local search

In a second set of experiments, we compared configurations that have the *local search* factor set to 2-opt. Considering that ACO algorithms using a local search component perform better than those not using one, experiments were carried out on a set of larger TSPLIB instances than in the previous case: pcb442, att532, rat783, u1060, nrw1379, d1655, and pr2392.

Cooperation was less beneficial for configurations adopting a 2-opt local search than for those adopting none, though RW and R were relatively less affected than FC and HC in the fixed-frequency case. Medium run results were more dependent on the *migration schedule* factor. Configurations using the fixed-frequency schedule obtained worse results more often than PIR when compared to configurations using the increasing-frequency schedule.

Table 3
The values in the table represent the number of times a configuration performed significantly better than PIR (numbers with a plus) and the number of times it performed significantly worse than PIR (numbers with a minus), according to the *two-sided pairwise Wilcoxon rank sum test*. Positive entries are integers in $[+1, +7]$, while negative entries are integers in $[-7, -1]$. Only 8 CPU configurations are listed.

LS	Migration frequency	Communication strategy	Run length		
			Short	Medium	Long
none	Fixed	FC	+5 −0	+4 −1	+4 −2
		HC	+6 −0	+4 −1	+3 −1
		RW	+6 −0	+4 −1	+4 −0
		R	+6 −0	+5 −1	+4 −2
	Increasing	FC	+0 −0	+4 −1	+3 −1
		HC	+0 −0	+5 −1	+4 −1
		RW	+0 −0	+2 −1	+4 −0
		R	+0 −0	+2 −0	+4 −0

Table 4
Numbers with a plus indicate a configuration performed significantly better than PIR; a minus indicates performance significantly worse than PIR. See Table 3 for a complete description.

LS	Migration frequency	Communication strategy	Run length		
			Short	Medium	Long
2-opt	Fixed	FC	+2 −0	+2 −1	+1 −4
		HC	+1 −0	+2 −1	+1 −4
		RW	+5 −0	+3 −0	+1 −0
		R	+4 −0	+3 −0	+1 −1
	Increasing	FC	+0 −0	+2 −1	+1 −3
		HC	+0 −0	+2 −1	+1 −3
		RW	+0 −0	+2 −0	+1 −0
		R	+0 −0	+2 −0	+2 −1

The best policy when using 2-opt seems to be dependent on the computational effort. For short runs, the replace-worst strategy with a fixed-frequency migration schedule should be used; for long runs, the replace-worst strategy with an increasing-frequency migration schedule be used instead. For medium runs, the best policy depends on the instance.

7.3. Configurations with 3-opt local search

The configurations using 3-opt local search were compared in a third set of experiments. To match the increase in performance of the algorithms when compared to those using 2-opt, the experiments were carried out on even larger TSPLIB instances: rat783, ul060, nrw1379, dl655, pr2392, fnl4461, and rl5915.

For all the configurations, there was a reduction in the number of times that configurations performed better or worse than PIR with respect to the 2-opt and none local search cases (Tables 4 and 5). As with the 2-opt and none local

Table 5
Numbers with a plus indicate a configuration performed significantly better than PIR; a minus indicates performance significantly worse than PIR. See Table 3 for a complete description.

LS	Migration frequency	Communication strategy	Run length		
			Short	Medium	Long
3-opt	Fixed	FC	+2 −0	+1 −2	+1 −3
		HC	+0 −0	+1 −1	+1 −2
		RW	+3 −0	+2 −0	+1 −0
		R	+3 −0	+2 −0	+1 −2
	Increasing	FC	+1 −0	+2 −0	+1 −0
		HC	+0 −0	+1 −0	+1 −0
		RW	+1 −0	+1 −0	+1 −0
		R	+0 −0	+1 −0	+1 −0

Table 6
ANOVAs were performed on the configurations for each instance at each run length of each local search level. The results are summarized in this table showing the (usually) best performing migration frequency and communication strategy (not including PIR). Exceptions to the listed overall trends are noted. ANOVA assumptions were violated in many cases, and those results are excluded from this summary. For configurations with no local search, ANOVA assumptions were violated on: kroA100 for all run lengths; att532 for medium runs; and kroA200 and pb442 for long runs. For configurations with 2-opt local search, ANOVA assumptions were violated on: pcb442 for all run lengths; att532 for medium and long runs; and rat783 for long runs. For configurations with 3-opt local search, ANOVA assumptions were violated on rat783, ul060, and dl655 for all run lengths, and pr2392 for long runs.

LS	Run length	Migration frequency	Communication strategy	Exceptions
none	Short	Fixed	FC, HC	HC (not FC) for eil101
	Medium	Fixed	FC	FC increasing for eil101 RW for att532
	Long	Increasing	FC, HC	R for eil101 Fixed for pcb442
2-opt	Short	Fixed	HC, R	HC for nrw1379 and larger R for ul060 and smaller
	Medium	Fixed	RW	
	Long	Increasing	RW, R	Fixed for RW R for pr2392 and larger
3-opt	Short	Fixed	RW, R	FC for fnl4461
	Medium	Fixed	RW	R for fnl4461
	Long	Increasing	R	FC for fnl4461

search cases, even the small beneficial effects of cooperation in the `3-opt` case tend to be reduced with increasing computational effort.

For medium and long runs, configurations adopting the fixed-frequency schedule more often performed worse than `PIR`. Overall, the best policy seems to be the replace-worst policy with an increasing-frequency schedule.

7.4. Summary of all local search configurations

For each set of local search configurations, we performed ANOVA analyses considering the results for short, medium, and long runs separately for each of the seven TSPLIB instances used. The set of instances chosen corresponded to the strength of the local search, with smaller instances for no local search and larger instances for stronger local search. As described in Section 6, before applying an ANOVA to a given set of data, we must check the validity of the key assumptions for the

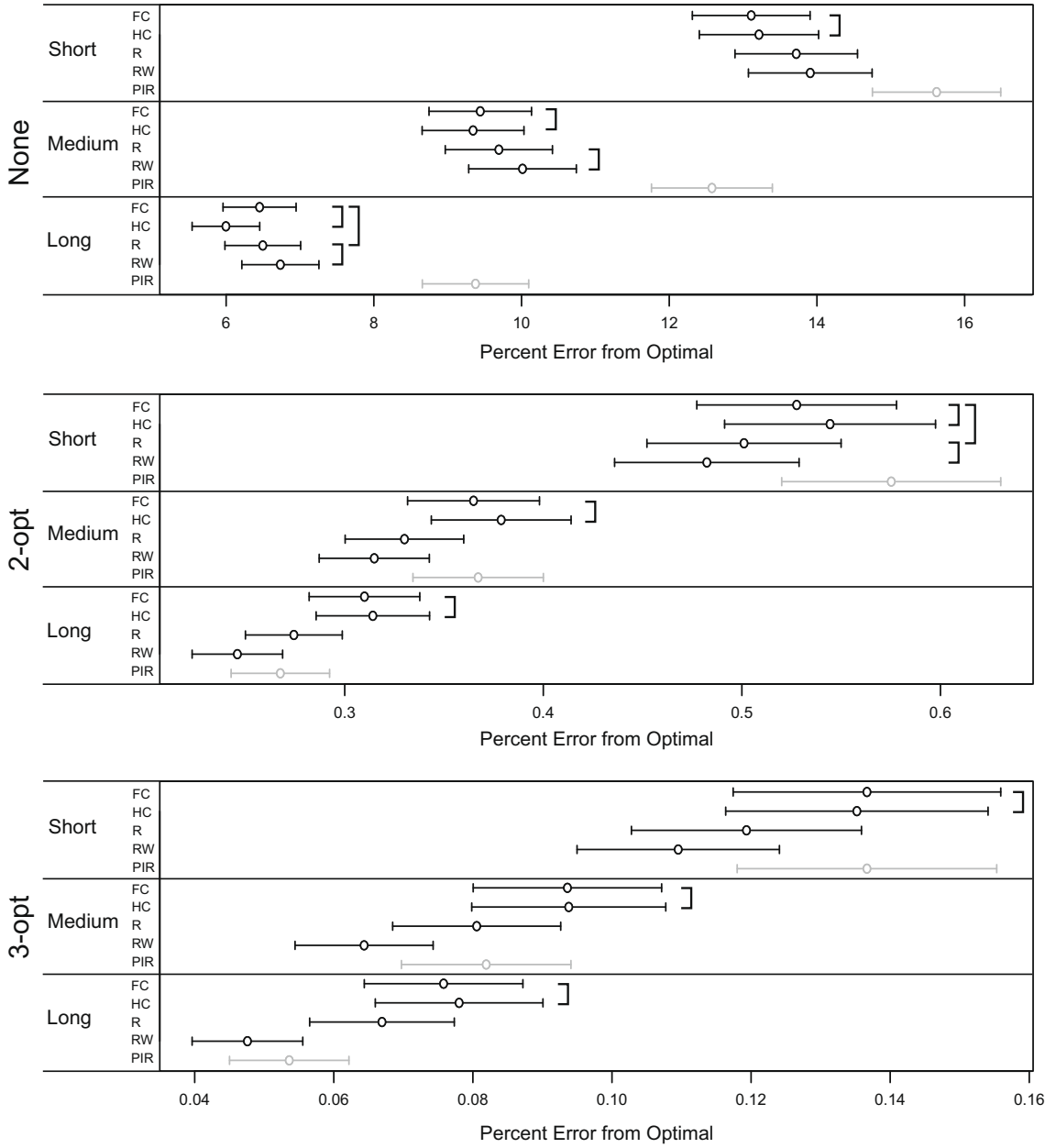

Fig. 3. The average percent distance from the optimal solution, with a 95% confidence interval, of each communication strategy (in order from top to bottom: `FC`, `HC`, `R`, `RW`, and `PIR`, with `PIR` highlighted in light grey), split by run length and local search strength, and averaged across all instances tested. Results are shown for the eight colonies, fixed frequency migration configurations only. Configurations (excluding `PIR`) not linked by brackets were statistically significantly different from each other using a permutation test with *p-values* adjusted by Holm's method.

fixed-effects model (homoscedasticity, independence of residuals, and normality of residuals). The results of the ANOVA analyses, as well as instances which were excluded due to violated assumptions, are summarized in Table 6. Overall, the ANOVAs indicate that the best performing communication strategies without local search are not the same as the strategies with increasing levels of local search. However, and especially when factoring in the number of instances excluded and the number of exceptions to the trends found, the trends shown by Table 6 would benefit from the collection of additional data.

This was done for the 8 colonies case, sampling across a broader range of problem instances to help determine the consistency of the trends seen earlier. In the additional data, the trends remained the same. The amount that communication benefited each of the configurations varied by the strength of the local search and the length of the run (see Fig. 4 for a summary of all the configurations by local search level). In general, and especially for the fixed case, the stronger the local search the less helpful increasing amounts of communication become for longer runs on smaller instances. In Fig. 3, which shows a representative fraction of the results (8 colonies, fixed frequency migration schedule configurations), it is clear that, in the absence of local search, the communication strategy plays an important role in achieving high solution quality. That role is diminished by increasing levels of local search and longer run lengths. Moreover, increased levels of local search changes which communication strategies perform the best.

Statistical significance of differences in performance between communication strategies was determined by means of a permutation test for each local search level, run length, and combination of two communication strategies, with *p-values* adjusted by Holm's method. Data labels were exchanged within blocks determined by the instance that data was collected on (e.g., FCf0 data points from the short run case on instance kroA100 were exchanged with RWf0 data points from the short run case on instance kroA100 as part of the comparison between FC and RW for short runs with no local search). Without local search, FC and HC policies typically performed better than both R and RW. However, when including local search, the HC and FC policies typically perform statistically significantly worse than RW, and, at the highest level of local search, worse than both RW and R. This is consistent with the trends found earlier (Table 6), and confirms that the best performing communication strategy changes depending on the level of local search used.

8. Scalability test

In a fourth set of experiments, the most successful policies identified previously were tested with three more levels of the *number of colonies* factor: 16, 32, and 64. In Section 7 we concluded that the overall best *communication strategies* are the replace-worst and the unidirectional ring when including a local search component. We also showed that the increasing-frequency schedule has beneficial effects that tend to be more noticeable for 8 colonies rather than for 4. This fourth set of experiments was carried out on three TSPLIB instances (nrw1379, pr2392, and rl5915) with 20 runs of 10,000 iterations each. As can be seen in Table 7, the beneficial effects of communication are still apparent even though their magnitude depends on both the length of the search and the level of the *local search* factor. Both RWi and Ri have a better average performance compared to PIR consistently over all the levels of the *number of colonies* factor in the long run case. While the RWi strategy still performs well overall, it performs worse than Ri when dealing with 16, 32 and 64 colonies. The greater the computational effort, the more evident the effect.

Table 7
The values in the table represent the number of times the communication policy in the row performed significantly better than parallel independent runs (numbers with a plus) and the number of times it performed significantly worse than parallel independent runs (numbers with a minus), according to the *two-sided pairwise Wilcoxon rank sum test*. Positive entries are integers in $[+1, +3]$, while negative entries are integers in $[-3, -1]$.

LS	Communication strategy	Run length					
		Medium			Long		
		16	32	64	16	32	64
2-opt	RWi	+1 −0	+1 −0	+1 −0	+1 −0	+1 −0	+0 −0
	Ri	+2 −0	+2 −0	+3 −0	+3 −0	+3 −0	+3 −0
3-opt	RWi	+0 −0	+1 −0	+0 −0	+2 −0	+1 −0	+1 −0
	Ri	+1 −0	+1 −0	+0 −0	+2 −0	+2 −0	+2 −0

Table 8
The symbols in the table indicate the performance of the communication policy in the row with respect to parallel independent runs for the tested instances of size as in the column. A plus (+) means that it has performed significantly better, according to the *two-sided pairwise Wilcoxon rank sum test*.

Configuration	Run length								
	Short			Medium			Long		
	316	1000	3162	316	1000	3162	316	1000	3162
8RWf2	+	+	+	+		+			+
8Rf2	+	+	+			+			+

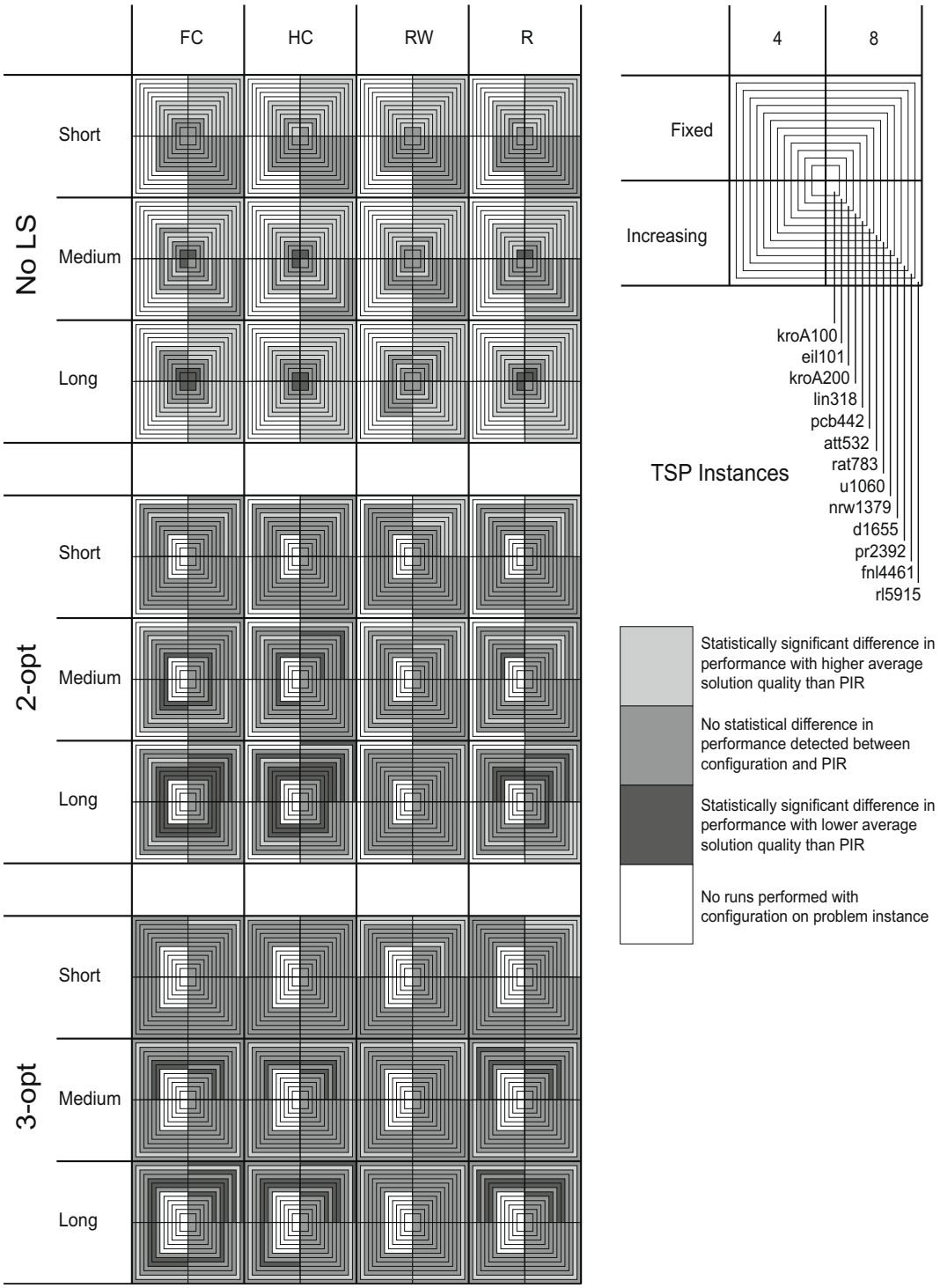

Fig. 4. Each stack of increasingly larger boxes represents a set of increasingly larger problem instances. Each stack is subdivided into four parts by the number of colonies (the columns) and the migration schedule (the rows). The stacks themselves are grouped by the local search amount, the run length, and the strategy employed. When a particular configuration performs better than `PIR` on a particular instance, the corresponding box is colored light grey. It is dark grey if the configuration performed worse, and medium grey otherwise. If the instance was not tested for that configuration, its coloration is left blank (white).

9. Test on unseen instances with 2-opt local search

To lend additional support for the conclusion of Section 8—that the cooperation of multiple colonies becomes less effective for increasing search length—we conducted a set of experiments on randomly generated traveling salesman problem instances. We generated several instances of different sizes where cities are distributed uniformly in a square: 30 instances of size 316, 30 instances of size 1000, and 30 instances of size 3162. On each instance, 1 run of 10,000 iterations was performed. The results are grouped by instance size and were analyzed as if they were a single instance with multiple runs.[7] To assess the statistical significance of the differences in solution costs obtained by the algorithms under analysis, we rely on a *two-sided pairwise Wilcoxon rank sum test* with *p-values* adjusted by Holm's method. To limit the computational budget, we restricted our test to the configurations `8RWf2` and `8Rf2`.

The results in Table 8 show that, as the run length is increased, both the `RWf` and `Rf` communication strategies are only statistically significantly better than `PIR` for the largest instances. This is consistent with our conjecture that the cooperation of multiple colonies becomes less effective for increasing search length and smaller instances.

10. Discussion and conclusions

Some preliminary studies on cooperation in multi-colony ACO algorithms, using `3-opt` local search and a fixed frequency migration schedule [29], led us to conjecture that information sharing becomes less effective for increasing search length and higher-performing algorithms. In this article, we have presented a study in which we have analyzed the impact that communication policies have on the solution quality reached by a parallel homogeneous multi-colony ACO algorithm for the traveling salesman problem. We adopted a full factorial design, empirically testing the configurations on a distributed-memory parallel architecture, and we have analyzed the results with a fixed-effects model ANOVA. We have considered several factors that influence the performance of a multi-colony ACO algorithm: the number of colonies, migration schedules, communication strategies on different interconnection topologies, and the usage of local search.

As a first step in the analysis of the experimental results, every configuration was compared against the most basic policy for multi-colony ACO algorithms—the parallel independent execution on multiple processors of the single colony algorithm—using the percentage error deviation from the known optimal cost. Our analysis suggests that, for the considered instances, the best communication policy for ACO algorithms that do not use local search are very different from those where the ACO algorithm makes use of a local search procedure. This indicates three things. First, studying communication policies in multi-colony ACO algorithms for the traveling salesman problem without local search and then simply adding the local search component leads to suboptimal performance. Second, the replace-worst (`RW`) and the unidirectional ring (`R`) seem to be the best performing strategies for multi-colony ACO algorithms using local search for the traveling salesman problem. And third, a variable increasing-frequency schedule to migrate solutions scales better with the search length and the number of colonies.

We can interpret these three conclusions as part of an exploitation/exploration tradeoff. Communication emphasizes exploitation by recruiting colonies to work in the same region of the search space. Less communication has an explorative effect, since each colony is more likely to be searching in a different part of the search space. Without local search, `PIR` appears to suffer from an overemphasis on exploration. Thus, any increase in exploitation results in improved solution quality, as when communication alone is added. As a result, the schemes that most strongly propagate the best solutions, as found by the fully-connected and hypercube topologies under the fixed migration frequency scheme typically perform the best. However, local search is also strongly exploitative, and coupling it alone with `PIR` can be expected to create a similarly more favorable balance of exploitation/exploration for problem sizes proportional to the amount of local search used. When adding local search and communication together, the algorithms tend to perform still better on larger problem instances, but poorer on smaller ones than `PIR` (see Fig. 4)—suffering from an overemphasis on exploitation in those cases. The smallest problems are generally solved with ease by algorithms that employ local search, and this mitigates the effect of any overemphasis on exploitation in those cases (such as for the instances `kroA100`, `eil101`, `kroA200`, `lin318`, and `pcb442` in this study). This is reflected by the fact that the communication schemes that less strongly propagate the best found solutions and leave the colonies to search more independently tend to perform better: the best performance is typically obtained by the ring topology or by the replace-worst scheme that use the increasing migration frequency scheme.

Since replace-worst is the closest in terms of communication to `PIR` (the colonies work independently, with the exception that one is updated with the current best so far solution), it is interesting that it typically never performs worse, and often times better than PIR. In contrast to fully-connected (`FC`), a small amount of communication can provide a modest improvement on larger problems with a minimum of over-exploitation on smaller problems. The ring strategy (`R`) is the next most similar strategy to `PIR` (in that it does not involve as many solution exchanges as `HC` or `FC`) and it has the next best performance of the four communication strategies for the cases with local search. Thus it seems that when adding local search, a large amount of communication is not necessarily desirable because it tends to lead towards an overemphasis on exploitation at the expense of exploration in some instances.

[7] The optimal value for each instance has been obtained either by exactly solving the problem using the publicly available TSP solver CONCORDE or, for the few instances for which the optimal solution was not found after 24 h, taking the best solution found after 1 h by a publicly available implementation of the Iterated Lin-Kernigham Helsgaun algorithm available at the URL: http://www.akira.ruc.dk/~keld/research/LKH/index-1.3.html.

The fact that a variable increasing-frequency schedule tends to perform better than a fixed frequency schedule, especially for larger runtimes, indicates that changing the ratio of exploitation to exploration over time is a better strategy than a fixed ratio. Shifting the emphasis in the course of a run from exploration to exploitation may initially delay finding high solution qualities (compare the short and medium run lengths to the long run lengths for the increasing strategy in Fig. 4), but it results in a more consistent final improvement over PIR in the long run.

This study demonstrates that the cooperation of multiple homogenous colonies becomes less effective for increasing search effort and stronger local search algorithms. Additionally, and most importantly for subsequent studies, it shows that the relative effectiveness of different communication policies changes with the addition of local search or, implying more generally, with the overall strength of the algorithm. As such, local search should be considered by studies that aim to recommend one communication policy over another for multi-colony ACO algorithms.

Acknowledgments

This work is supported by the 'COMP^{2}SYS,' 'ANTS,' and 'Meta-X' projects. 'COMP^{2}SYS' is a Marie Curie Early Stage Training Site, funded by the European Community's Sixth Framework Programme under contract number MEST-CT-2004-505079. The 'ANTS' and 'Meta-X' are Action de Recherche Concertée projects funded by the Scientific Research Directorate of the French Community of Belgium. M. Dorigo, T. Stützle and M. Birattari acknowledge support from the Fund for Scientific Research of the French Community of Belgium (F.R.S.-FNRS). C. Twomey acknowledges support through the Fulbright Program by the Commission for Educational Exchange between the US, Belgium, and Luxembourg. The information provided is the sole responsibility of the authors and does not reflect the opinion of the sponsors. The European Community and the French Community are not responsible for any use that might be made of data appearing in this publication. The authors wish to thank Ruben Ruiz and Marco Chiarandini for their support during the statistical analysis phase.

References

[1] E. Alba, M. Tomassini, Parallelism and evolutionary algorithms, IEEE Transactions on Evolutionary Computation 6 (5) (2002) 443–461.
[2] T.C. Belding, The distributed genetic algorithm revisited, in: L.J. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA), Morgan Kaufmann, 1995, pp. 114–121.
[3] C. Blum, Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling, Computers and Operations Research 32 (6) (2005) 1565–1591.
[4] M. Bondanza, M. Bolondi, Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore, Master's Thesis, Politecnico di Milano, Facoltà di Ingegneria, Milano, Italy, 1993.
[5] E. Cantú-Paz, A survey of parallel genetic algorithms, Calculateurs Parallèles, Reseaux et Systems Repartis 10 (2) (1998) 141–171.
[6] E. Cantú-Paz, D.E. Goldberg, Efficient parallel genetic algorithms: theory and practice, Computer Methods in Applied Mechanics and Engineering 186 (2000) 221–238.
[7] L. Chen, C.F. Zhang, Adaptive parallel ant colony optimization, Parallel and Distributed Processing and Applications 3758 (2005) 275–285.
[8] S.-C. Chu, J.F. Roddick, J.-S. Pan, Ant colony system with communication strategies, Information Sciences 167 (1-4) (2004) 63–76.
[9] S.-C. Chu, J.F. Roddick, J.-S. Pan, C.-J. Su, Parallel ant colony systems, in: N. Zhong, Z.-W. Ras, S. Tsumoto, E. Suzuku (Eds.), Foundations of Intelligent Systems: 14th International Symposium, ISMIS 2003, Lecture Notes in Artificial Intelligence, vol. 2871, Springer-Verlag, Heidelberg, Germany, 2003, pp. 279–284.
[10] W.J. Conover, Practical Nonparametric Statistics, Third ed., John Wiley and Sons, New York, NY, USA, 1999.
[11] M. Craus, L. Rudeanu, Parallel framework for cooperative processes, Scientific Programming 13 (3) (2005) 205–217.
[12] P. Delisle, M. Krajecki, M. Gravel, C. Gagné, Parallel implementation of an ant colony optimization metaheuristic with OpenMP, in: Proceedings of the Third European Workshop on OpenMP (EWOMP'01), Barcelona, Spain, 2001.
[13] J.-L. Deneubourg, S. Aron, S. Goss, J.-M. Pasteels, The self-organizing exploratory pattern of the argentine ant, Journal of Insect Behaviour 3 (2) (1990) 159–168.
[14] K.F. Doerner, R.F. Hartl, S. Benkner, M. Lucka, Parallel cooperative saving based ant colony optimization-multiple search and decomposition approaches, Parallel Processing Letters 16 (3) (2006) 351–369.
[15] M. Dorigo, Parallel ant system: an experimental study, Unpublished manuscript, 1993.
[16] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the travelling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[17] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, USA, 2004.
[18] Marco Dorigo, Mauro Birattari, Thomas Stützle, Ant colony optimization: artificial ants as a computational intelligence technique, IEEE Computational Intelligence Magazine 1 (4) (2006) 28–39.
[19] Issmail Ellabib, Paul Calamai, Otman Basir, Exchange strategies for multiple ant colony system, Information Sciences 177 (5) (2007) 1248–1264.
[20] L.M. Gambardella, M. Dorigo, Ant colony system hybridized with a new local search for the sequential ordering problem, INFORMS Journal on Computing 12 (3) (2000) 237–255.
[21] L.M. Gambardella, E. Taillard, G. Agazzi, MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 63–76.
[22] A. Grama, A. Gupta, G. Karypis, V. Kumar, Introduction to Parallel Computing, second ed., Pearson-Addison Wesley, Harlow, UK, 2003.
[23] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.
[24] H. Hoos, T. Stützle, Stochastic Local Search: Foundations and Applications, Morgan Kaufmann, 2005.
[25] J. Kennedy, R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, IEEE Transactions on Systems, Man, and Cybernetics, Part C 36 (4) (2006) 515–519.
[26] O. Korb, T. Stützle, T.E. Exner, Application of ant colony optimization to structure based drug design, in: M. Dorigo, L.M. Gambardella, M. Birattari, A. Martinoli, R. Poli, T. Stützle (Eds.), Ant Colony Optimization and Swarm Intelligence, Fifth International Workshop, ANTS 2006, Lecture Notes in Computer Science, vol. 4150, Springer-Verlag, Berlin, Germany, 2006, pp. 247–258.
[27] F. Krüger, D. Merkle, M. Middendorf, Studies on a parallel Ant system for the BSP model. Unpublished manuscript, 1998.
[28] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy-Kan, D.B. Shmoys, The Traveling Salesman Problem, John Wiley and Sons, New York, NY, USA, 1985.
[29] M. Manfrin, M. Birattari, T. Stützle, M. Dorigo, Parallel ant colony optimization for the traveling salesman problem, in: M. Dorigo, L.M. Gambardella, M. Birattari, A. Martinoli, R. Poli, T. Stützle (Eds.), Ant Colony Optimization and Swarm Intelligence, Fifth International Workshop, ANTS 2006, Lecture Notes in Computer Science, vol. 4150, Springer-Verlag, Berlin, Germany, 2006, pp. 224–234.

[30] Max Manfrin, Mauro Birattari, Thomas Stützle, and Marco Dorigo, Communication policies for a parallel multi-colony ACO algorithm with identical colonies, in: E. Ridge, T. Stützle, M. Birattari, H.H. Hoos (Eds.), SLS-DS 2007: Doctoral Symposium on Engineering Stochastic Local Search Algorithms, IRIDIA, CoDE, Université Libre de Bruxelles, Berlin, Brussels, Belgium, 2007, pp. 58–62.
[31] Max Manfrin, Thomas Stützle, Mauro Birattari, Marco Dorigo, Extended empirical analysis: an anlysis of communication policies for homogeneous multi-colony ACO algorithm through experimental design, 2008. <http://iridia.ulb.ac.be/supp/IridiaSupp2008-007/>.
[32] R. Michel, M. Middendorf, An island model based ant system with lookahead for the shortest supersequence problem, in: A.E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature-PPSN V, Lecture Notes in Computer Science, vol. 1498, Springer-Verlag, Heidelberg, Germany, 1998, pp. 692–701.
[33] M. Middendorf, F. Reischle, H. Schmeck, Information exchange in multi-colony ant algorithms, in: J. Rolim, G. Chiola, G. Conte, L.V. Mancini, O.H. Ibarra, H. Nakano (Eds.), Parallel and Distributed Processing: 15 IPDPS 2000 Workshops, Lecture Notes in Computer Science, vol. 1800, Springer-Verlag, Heidelberg, Germany, 2000, pp. 645–652.
[34] M. Middendorf, F. Reischle, H. Schmeck, Multi-colony ant algorithms, Journal of Heuristics 8 (3) (2002) 305–320.
[35] D.A.L. Piriyakumar, P. Levi, A new approach to exploiting parallelism in ant colony optimization, in: International Symposium on Micromechatronics and Human Science (MHS) 2002, IEEE, 2002, pp. 237–243.
[36] M. Randall, A. Lewis, A parallel implementation of ant colony optimization, Journal of Parallel and Distributed Computing 62 (9) (2002) 1421–1432.
[37] G. Reinelt, TSPLIB—a traveling salesman problem library, ORSA Journal on Computing 3 (1991) 376–384.
[38] T. Stützle, Parallelization strategies for ant colony optimization, in: A.E. Eiben, T. Bäck, M. Schoenauer, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature-PPSN V, Lecture Notes in Computer Science, vol. 1498, Springer-Verlag, Heidelberg, Germany, 1998, pp. 722–731.
[39] T. Stützle, Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications, vol. 220, DISKI, Infix, Sankt Augustin, Germany, 1999.
[40] T. Stützle, H.H. Hoos, MAX–MIN ant system, Future Generation Computer System 16 (8) (2000) 889–914.
[41] E.-G. Talbi, O. Roux, O. Fonlupt, D. Robillard, Parallel ant colonies for the quadratic assignment problem, Future Generation Computer System 17 (4) (2001) 441–449.
[42] R. Tanese, Distributed genetic algorithms, in: J.D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms (ICGA), Morgan Kaufmann, 1989, pp. 434–439.
[43] Cheng-Fa Tsai, Chun-Wei Tsai, Ching-Chang Tseng, A new hybrid heuristic approach for solving large traveling salesman problem, Information Sciences 166 (1-4) (2004) 67–81.
[44] Zhongzhen Yang, Bin Yu, Chuntian Cheng, A parallel ant colony optimization algorithm for bus network optimization, Computer-Aided Civil and Infrastructure Engineering 22 (1) (2007) 44–55.
[45] M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo, Model-based search for combinatorial optimization: a critical survey, Annals of Operations Research 131 (2004) 373–395.