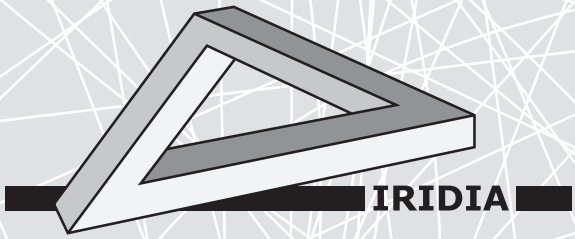


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Parallel Ant Colony Optimization for the Traveling Salesman Problem

Max Manfrin, Mauro Birattari,
Thomas Stützle, and Marco Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2006-007

March 2006

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2006-007

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsability for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

Parallel Ant Colony Optimization for the Traveling Salesman Problem

Max Manfrin, Mauro Birattari, Thomas Stützle, and Marco Dorigo

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

Abstract. There are two reasons for parallelizing a metaheuristic if interested in performance: (i) given a fixed time to search, the aim is to increase the quality of the solutions found in that time; (ii) given a fixed solution quality, the aim is to reduce the time needed to find a solution not worse than that quality. In this article, we study the impact of communication when we parallelize an high-performing ant colony optimization (ACO) algorithm for the traveling salesman problem using message passing libraries. In particular, we examine synchronous and asynchronous communications on different interconnection topologies. We find that the simplest way of parallelizing the ACO algorithms, based on parallel independent runs, is surprisingly effective; we give some reasons as to why this is the case.

1 Introduction

A system of n parallel processors is generally less efficient than a single n-times-faster processor, but the parallel system is often cheaper to build, especially if we consider clusters of PCs or workstations connected through fast local networks and software environments such as Message Passing Interface (MPI) [1, 2]. This makes clusters, currently, one of the most affordable, and therefore adopted, parallel architecture to develop parallel algorithms.

The availability of parallel architectures at low cost has widen the interest for the parallelization of algorithms and metaheuristics [3]. Some research has been done also on the parallelization of ant colony optimization (ACO) algorithms, but, surprisingly enough, only few works used as a basis for the parallelization a high-performing ACO algorithm. One such exception is [4], where the effect of parallel independent runs was studied.

When developing parallel population-based metaheuristics such as parallel genetic algorithms and parallel ACO algorithms, it is common to adopt the "island model" approach [5], in which the exchange of information plays a major role. Solutions, pheromone matrices, and parameters have been tested (see for example [6–9]) as the object of such exchange, but no convincing positive results have been achieved.

In this paper, we study how different interconnection topologies affect the overall performance when we want to increase, given a fixed run time, the quality of the solutions found by a multicolony parallel ACO algorithm to solve the

traveling salesman problem (TSP), an $\mathcal{NP}$-hard problem [10] commonly used as a test set when developing metaheuristics. For each interconnection topology, we implement both a synchronous and an asynchronous communication. The communication strategy we adopt involves the exchange of the best-so-far solutions every r iterations, after an initial period of "solitary" search. An advantage of using best-so-far solutions over pheromone matrices is that less data has to be exchanged: for the smallest instance that we consider, each pheromone matrix requires several megabytes of memory space, while a solution requires only some kilobytes.

The article is structured as follows. Section 2 describes the details of the ACO algorithm we use as a basis for parallelization, and describes the different interconnection topologies adopted. In Section 3, we report details about the experimental setup, and Section 4 contains the results of the computational experiments. Finally, in Section 5 we discuss the limitations of this work and summarize the main conclusions that can be drawn from the experimental results.

2 Parallel Implementation of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

Ant Colony Optimization (ACO) is a metaheuristic introduced in 1991 by Dorigo and co-workers [11, 12]. For an overview of the currently available ACO algorithms see [13]. In ACO, candidate solutions are generated by a set of stochastic procedures called artificial ants that use a parametrized probabilistic model which is updated using the previously seen solutions [14].

For this research, we use $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [15]—currently one of the best-performing ACO variants—as a basis for our parallel implementation. To have a version that is easily parallelizable, we removed the occasional pheromone re-initializations applied in the $\mathcal{MM}$AS described in [15], and we use only a best-so-far pheromone update. Our implementation of $\mathcal{MM}$AS is based on the publicly available ACOTSP code, but it extends it by quadrant nearest neighbor lists. Our version also includes a 3-opt local search.

We aim at an unbiased comparison of the performance produced by communication among multiple colonies on five different interconnection topologies. In order to obtain a fair and meaningful analysis of the results, we have restricted the approaches to the use of a constant communication rate among colonies to exchange the best-so-far solutions. A colony *injects* in his current solution-pool a received best-so-far solution if and only if it is better than its current best-so-far solution, otherwise it disregards it. In the following, we briefly and schematically describe the principles of the communication on each interconnection topology we considered. For each topology, with the exception of the Parallel independent runs, we have two versions: a first one, where the communication is synchronous, and a second one, where the communication is asynchronous.

Completely-connected. In this parallel model, k colonies communicate with each other and cooperate to find good solutions. One colony acts as a *master* and

collects the values of the best-so-far solutions found by the other $k-1$ colonies. The *master* then broadcasts to all colonies the identifier of the colony that owns the best solution among all k colonies so that everybody can get a copy of this solution. In Section 4 and 5, we identify the two implementations of this model with the acronym SCC for the synchronous completely-connected and ACC for the asynchronous one.

Replace-worst. This parallel model is similar to Completely-connected, with the exception that the *master* identifies also the colony that owns the worst solution among the k colonies. Instead of broadcasting the identity of the best colony, the *master* send only one message to the best colony containing the identity of the worst colony. In this way, the best colony send its best-so-far solution only to the worst colony. In Section 4 and 5, we identify the two implementations of this model with the acronym SRW for the synchronous replace-worst and ARW for the asynchronous one.

Hypercube. In this parallel model, k colonies are connected according to the hypercube topology (see [16] for a detailed explanation of it). Practically, each colony is located on a vertex i of the hypercube and can communicate only with these colonies that are located in the vertices that are directly connected to i. Each colony sends to each of its neighbors its best-so-far solution. In Section 4 and 5, we identify the two implementations of this model with the acronym SH for the synchronous hypercube and AH for the asynchronous one.

Ring. Here, k colonies are connected in such a way that they create a ring. For this research we have implemented a unidirectional ring, so that colony i sends his best-so-far solution only to colony $[(i+1) \bmod k]$, and receives it only from colony $[(i-1+k) \bmod k]$. In Section 4 and 5, we identify the two implementations of this model with the acronym SR for the synchronous ring and AR for the asynchronous one.

Parallel independent runs. In this parallel model, k copies of the same sequential $\mathcal{MM}$AS algorithm are simultaneously and independently executed using different random seeds. The final result is the best solution among all the obtained ones. Using parallel independent runs is appealing as basically no communication overhead is involved and nearly no additional implementation effort is necessary. In Section 4 and 5, we identify the implementation of this model with the acronym PIR.

These topologies allow us to consider decreasing communication volumes, moving from more global communication, as in Completely-connected, to more local communication, as in Ring, to basically no communication, as in Parallel independent runs.

3 Experimental Setup

All algorithms are coded in C using MPI libraries under the same development framework. Experiments were performed on a homogeneous cluster of 4 computational nodes running GNU/Linux Debian 3.0 as Operating System and LAM/MPI 7.1.1 as communication libraries. Each computational node contains two AMD Opteron™ 244 CPUs, 2 GB of RAM, and one 1 Gbit Ethernet network card. The nodes are interconnected through a 48-ports Gbit switch.

Computational experiments are performed with $k = 8$ colonies of 25 ants each that exchange the best-so-far solution every 25 iterations, except for the first 100 iterations.

We consider 10 instances from the TSPLIB [17] with a termination criterion based on run time, dependent on the size of the instance, as reported in Table 1. For each of the 10 instances, 10 runs were performed. In order to have a reference algorithm for comparison, we also test the equivalent sequential $\mathcal{MM}$AS algorithm. In the first case (SEQ), it runs for the same overall wall-clock time as a parallel algorithm (8-times the wall-clock time of a parallel algorithm), while in the second case (SEQ2), it runs for the same wall-clock time as one CPU of the parallel algorithm.

The parameters of $\mathcal{MM}$AS are chosen in order to guarantee robust performance over all the different sizes of instances; we use the same parameters as proposed in [15], except for the pheromone re-initializations and the best-so-far update, as indicated in Section 2.

To compare results across different instances, we normalize them with respect to the distance from the known optimal value. For a given instance, we denote as c_{MH} the value of the final solution of algorithm MH, and c_{opt} the value of the optimal solution; the normalized value is then defined as

$$\text{Normalized Value for MH} = \frac{c_{\mathrm{MH}} - c_{\mathrm{opt}}}{c_{\mathrm{opt}}} \cdot 100. \tag{1}$$

Table 1. Instances with run time in seconds and average number of total iterations in a run done by the sequential algorithm SEQ2

instance	run time	SEQ2 average iterations
pr1002	900	11831
u1060	900	10733
pcb1173	900	10189
d1291	1200	11325
nrw1379	1200	8726
fl1577	1500	15938
vm1748	1500	6160
rl1889	1500	6199
d2103	1800	12413
pr2392	1800	8955

Table 2. *p-values* for the null hypothesis "The distribution of the % distance from optimum of solutions for all instances is the same as PIR". The alternative hypothesis is that "The median of PIR distribution is lower". The significance level with which we reject the null hypothesis is 0.05

SCC	ACC	SRW	ARW	SH	AH	SR	AR
5.4e-4	0.01	0.02	0.02	1.2e-3	0.02	0.02	0.02

This normalization method provides a measure of performance that is independent of the different instance hardness. It is reasonable to request that a parallel algorithm performs at least no worse than SEQ within the computation time under consideration.

4 Results

As stated in Section 2, we aim at an unbiased comparison of the performance produced by communication among multiple colonies on different interconnection topologies. The hypothesis is that the exchange of the best-so-far solution among different colonies speeds up the search of high quality solutions, having a positive impact on the performance of the algorithms. In order to test the advantages of communication, we implement two versions of each algorithm: a first one where the communication is synchronous, and a second one where the communication is asynchronous. This setup allows us to use statistical techniques for verifying if differences in solutions quality found by the algorithms are statistically significant. Figure 1 contains the boxplot of the results[1] grouped by algorithm over all instances after the normalization described in Section 3. The boxplot indicates that, on average, all the parallel models, except SCC, seem able to do better than SEQ and SEQ2, but that the best performing approach is PIR. We check whether the differences in performance among the parallel models with exchange of information and PIR are statistically significant. The assumptions for a parametric method are not met, hence we rely on the *Wilcoxon rank sum* test [18] with *p-values* adjusted by Holm's method [19].

The differences in performance of all the parallel models with information exchange from those of PIR are statistically significant, as can be seen from Table 2, confirming that PIR is the best performing approach under the tested conditions. We also check whether the differences in performance are statistically significant once we group the algorithms by parallel model, using again the Wilcoxon test with p-values adjusted by Holm's method, reporting the results in Table 3. All the algorithms have differences in performance that are not statistically significant.

[1] We refer the reader interested in the raw data to the URL:
`http://landau.ulb.ac.be/~mmanfrin/papers/ants06/ants06.html`

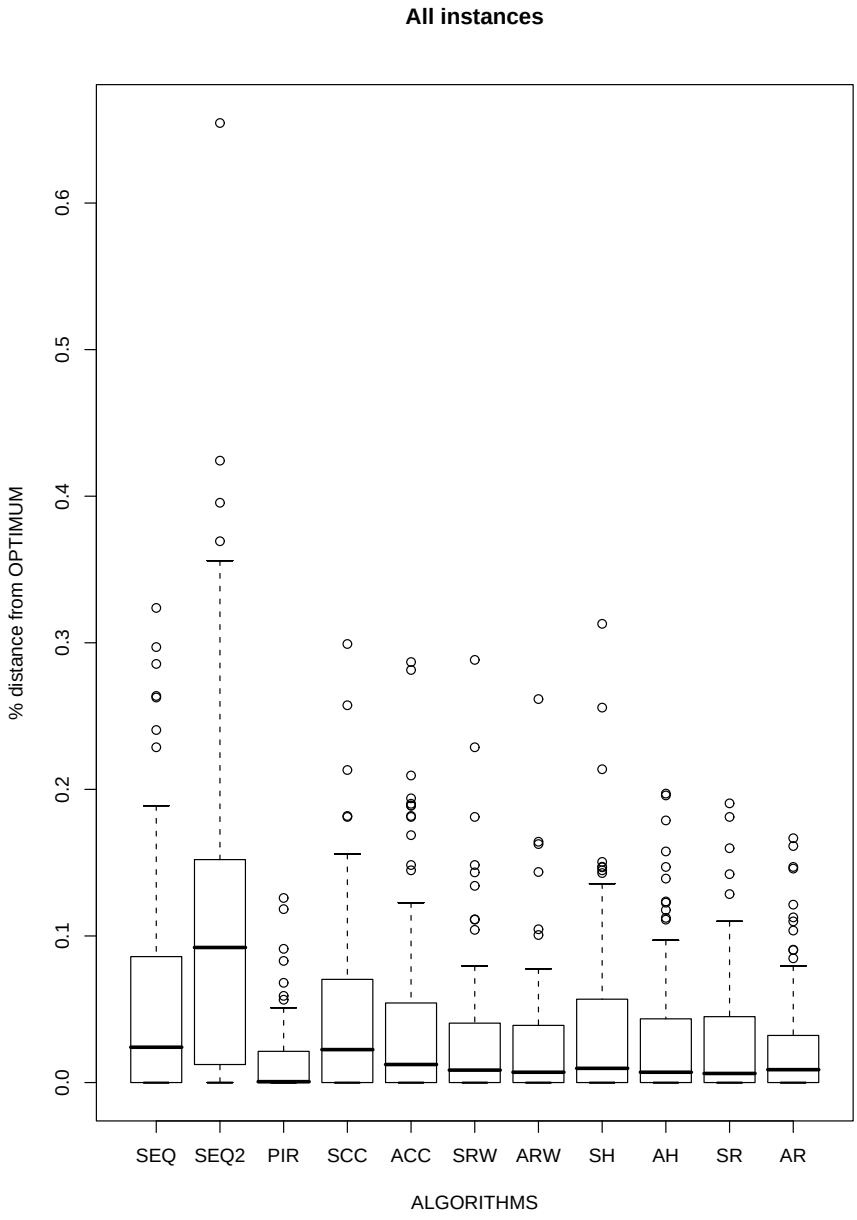


Fig. 1. Aggregate results over all instances. Boxplot of normalized results

Table 3. *p-values* for the null hypothesis "The distributions of the % distance from optimum of solutions are the same" for all instances. The significance level with which we reject the null hypothesis is 0.05

	CC	**RW**	**H**
RW	0.55	-	-
H	1	1	-
R	0.55	1	1

Even though all parallel algorithms achieve better performance than the sequential ones, these are negative results for the parallel models that rely on communication. One reason might be that the run times are rather high, and $\mathcal{MM}$AS easily converges in those times. This could bias the performance in favor of PIR, that can count on multiple independent search paths to explore the search space, while the other parallel algorithms tend to converge too fast toward a same solution, due to the frequent exchange of information (this is verified by the traces of the algorithms' outputs).

In order to check if our doubt on the "stagnation" behavior has some fundament, we re-analyze the results considering a reduced run time by a factor $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}$, showing the resulting boxplots in Figures 2, 3, 4, 5, 6, 7, 8. We observe that the more we reduce the run time, the less the differences are between the performance of the SEQ algorithm and the others, up to the reduction factor of $\frac{1}{64}$, for which SEQ performs in average better than all the parallel models (please, remember that SEQ has a run time that is 8-times the one of the other parallel algorithms).

To further confirm the "stagnation" behavior, we analyze the run-time distribution (RTD) of the sequential algorithm for reaching the known optimal solution value. In Figure 9 we give plots of the measured RTDs for the two instances pr1002 and d2103. As explained in [20], the exponential distribution that is given in these plots indicates that this version of $\mathcal{MM}$AS may profit strongly from algorithm restarts and, hence, this is an indication of severe stagnation behavior.

To better understand the impact that the frequency of communication has on performance, we change the communication scheme from the exchange of the best-so-far solutions every 25 iterations, except for the first 100, to the exchange every $\frac{n}{4}$ iterations, except during the first $\frac{n}{2}$, where n is the size of the instance. Given the previous results, according to which there are no statistically significant differences in the performance of the parallel algorithms, we test this new communication scheme on the parallel models Replace-worst (SRW2) and Ring (SR2). Figure 10 contains the boxplots of the results. Once more, we rely on the Wilcoxon test with Holm's adjustment (see Table 4) to verify if the differences in performance are statistically significant.

With the adoption of the new communication scheme, under the same experimental conditions, we are not able to reject the null hypothesis "The distribu-

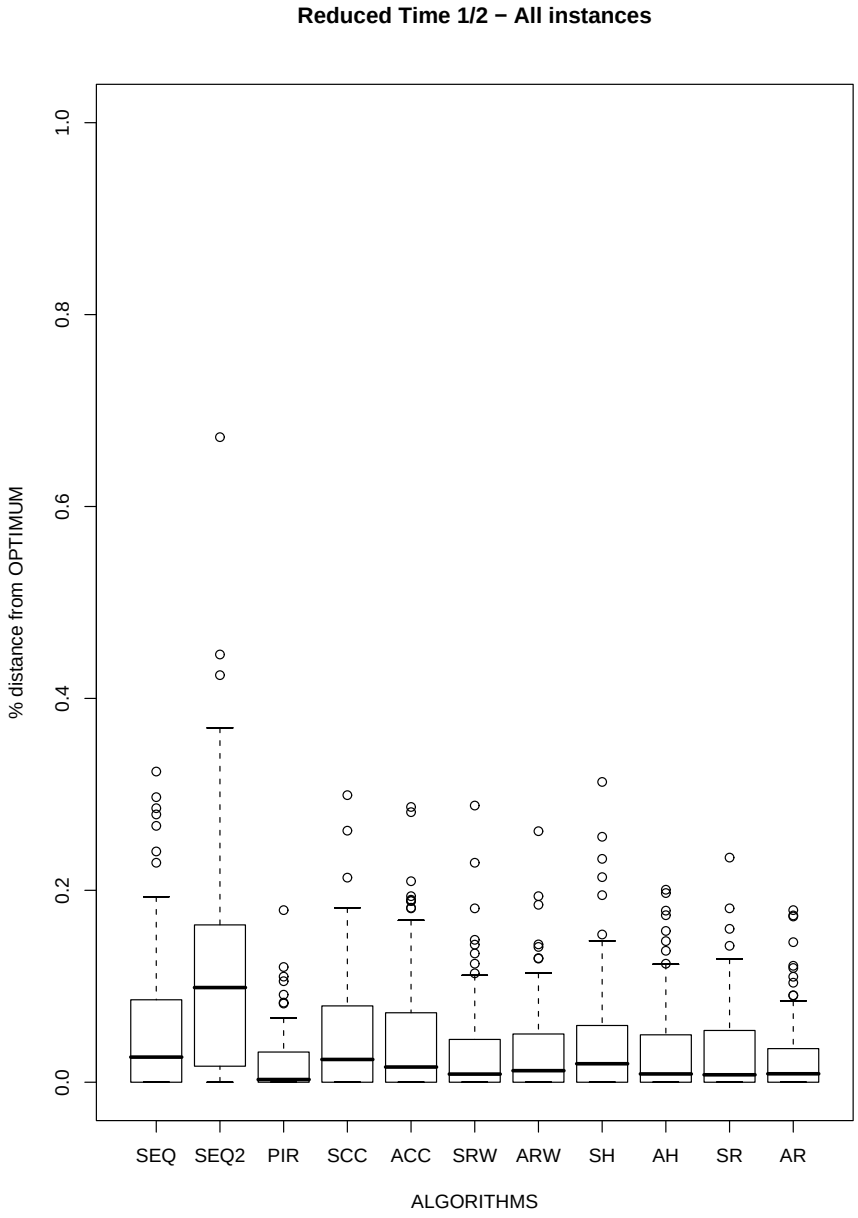


Fig. 2. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. Run time reduced by factor $\frac{1}{2}$

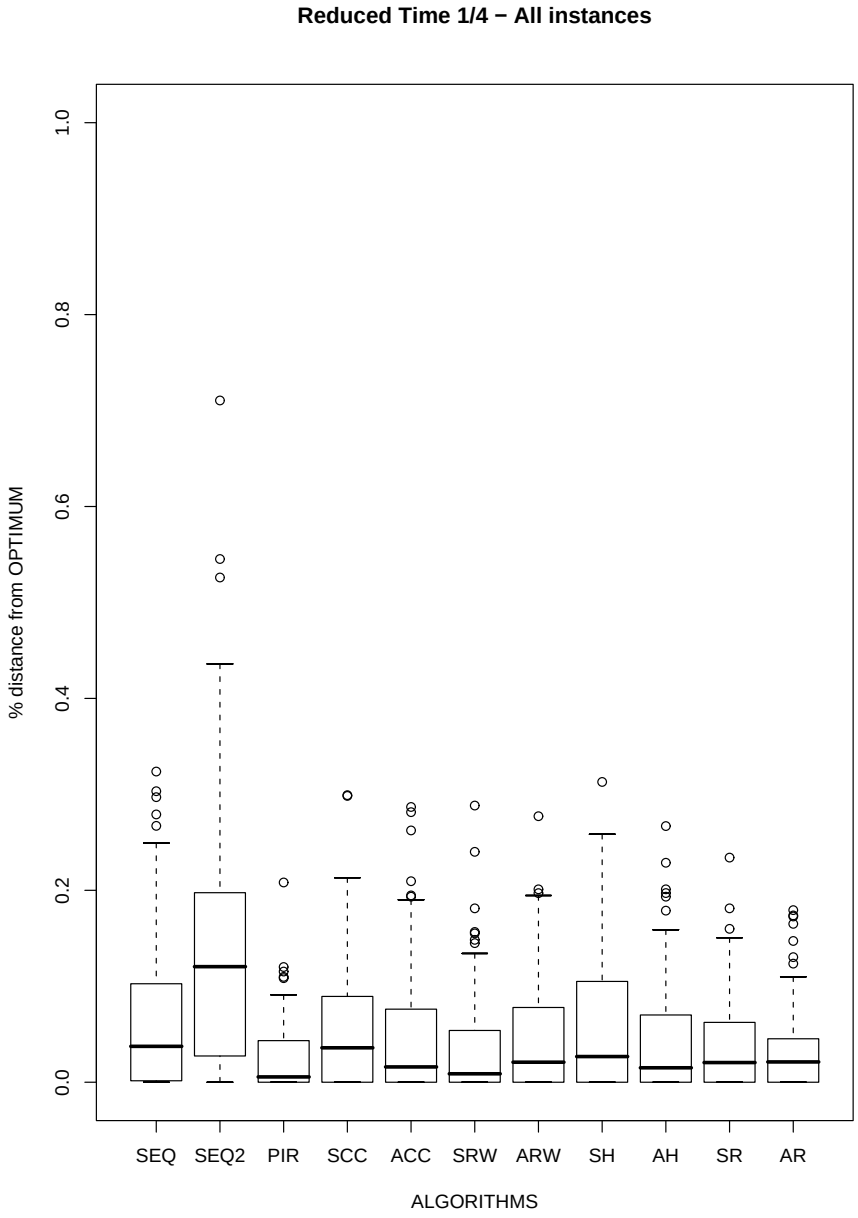


Fig. 3. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. Run time reduced by factor $\frac{1}{4}$

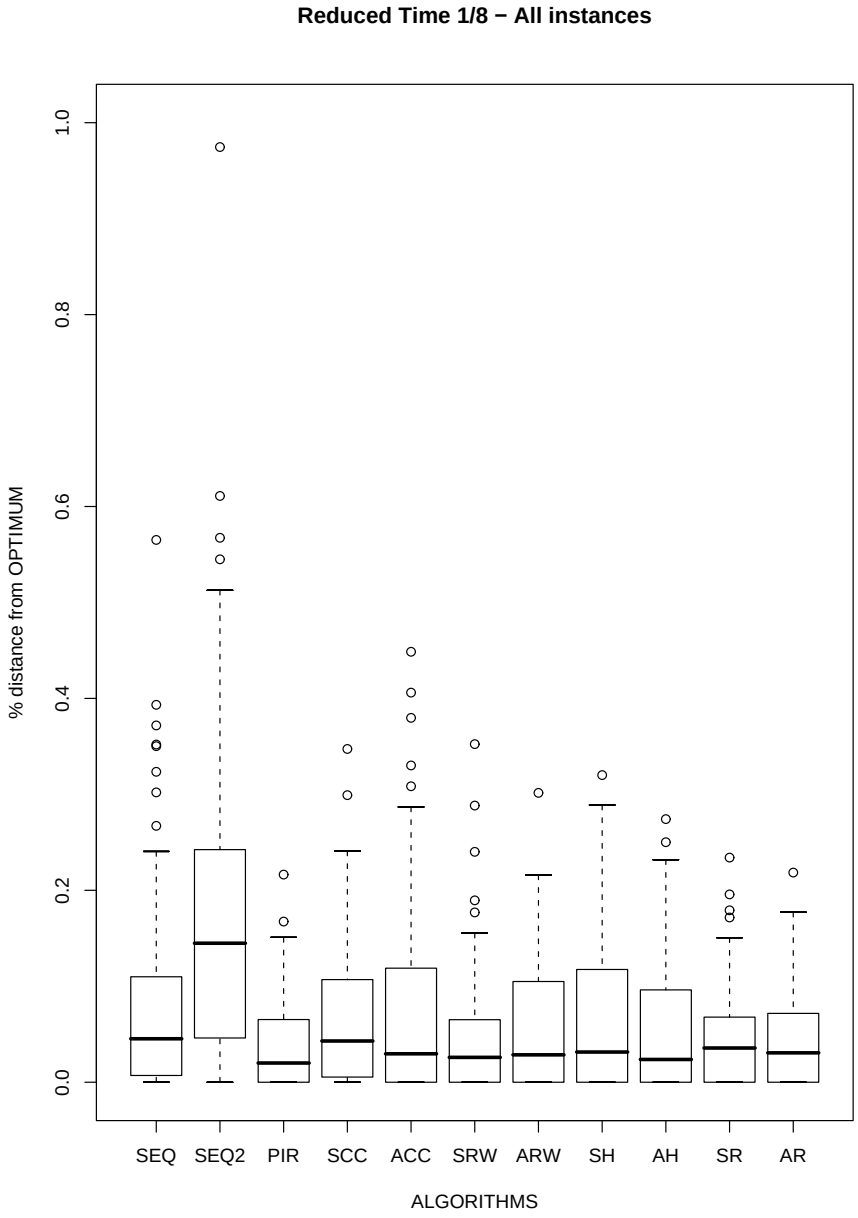


Fig. 4. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. Run time reduced by factor $\frac{1}{8}$

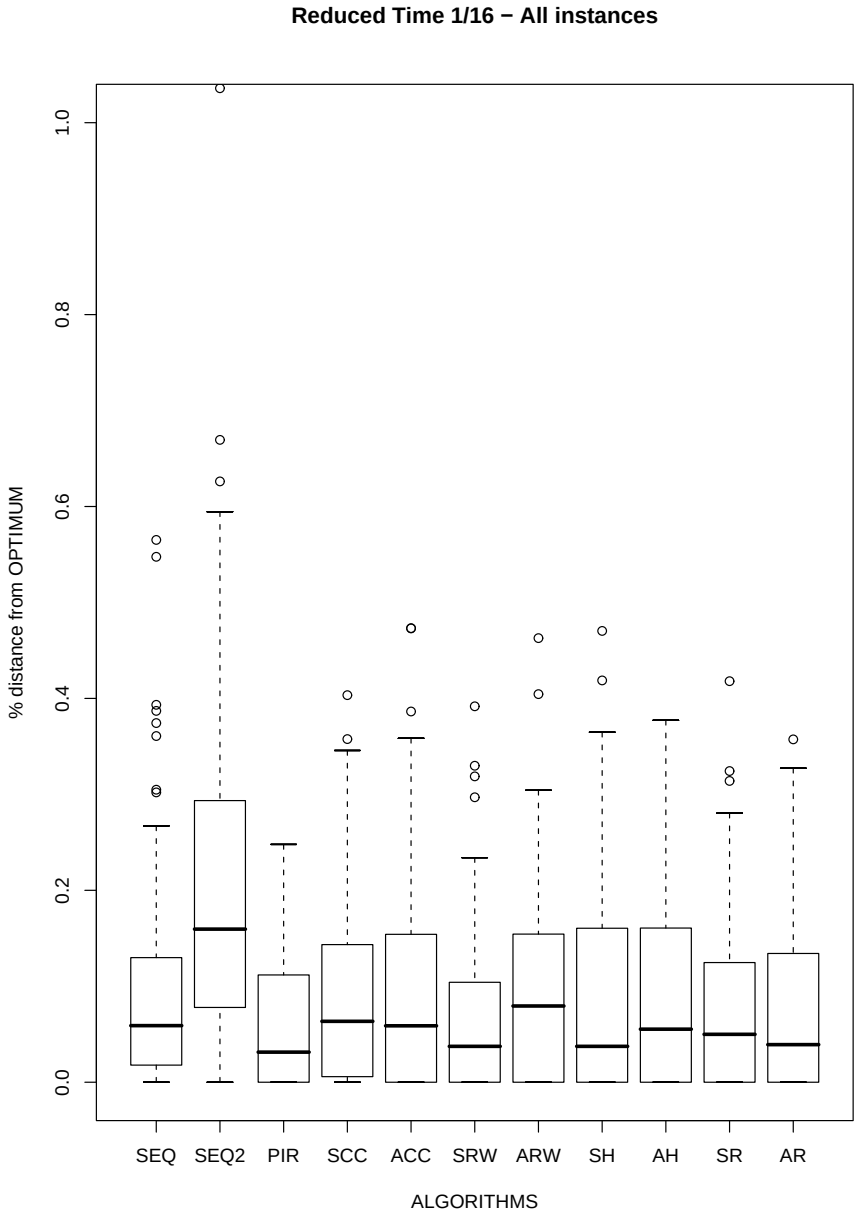


Fig. 5. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. Run time reduced by factor $\frac{1}{16}$

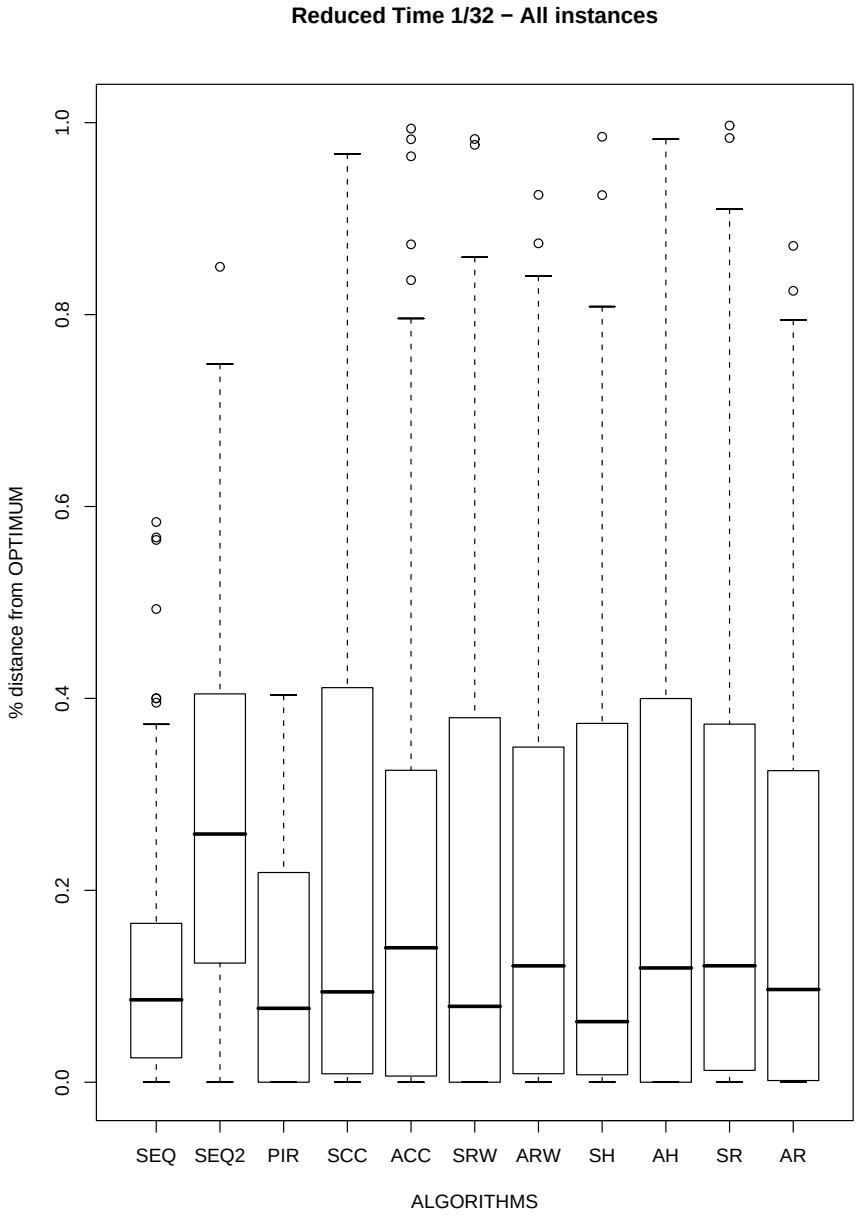


Fig. 6. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. Run time reduced by factor $\frac{1}{32}$

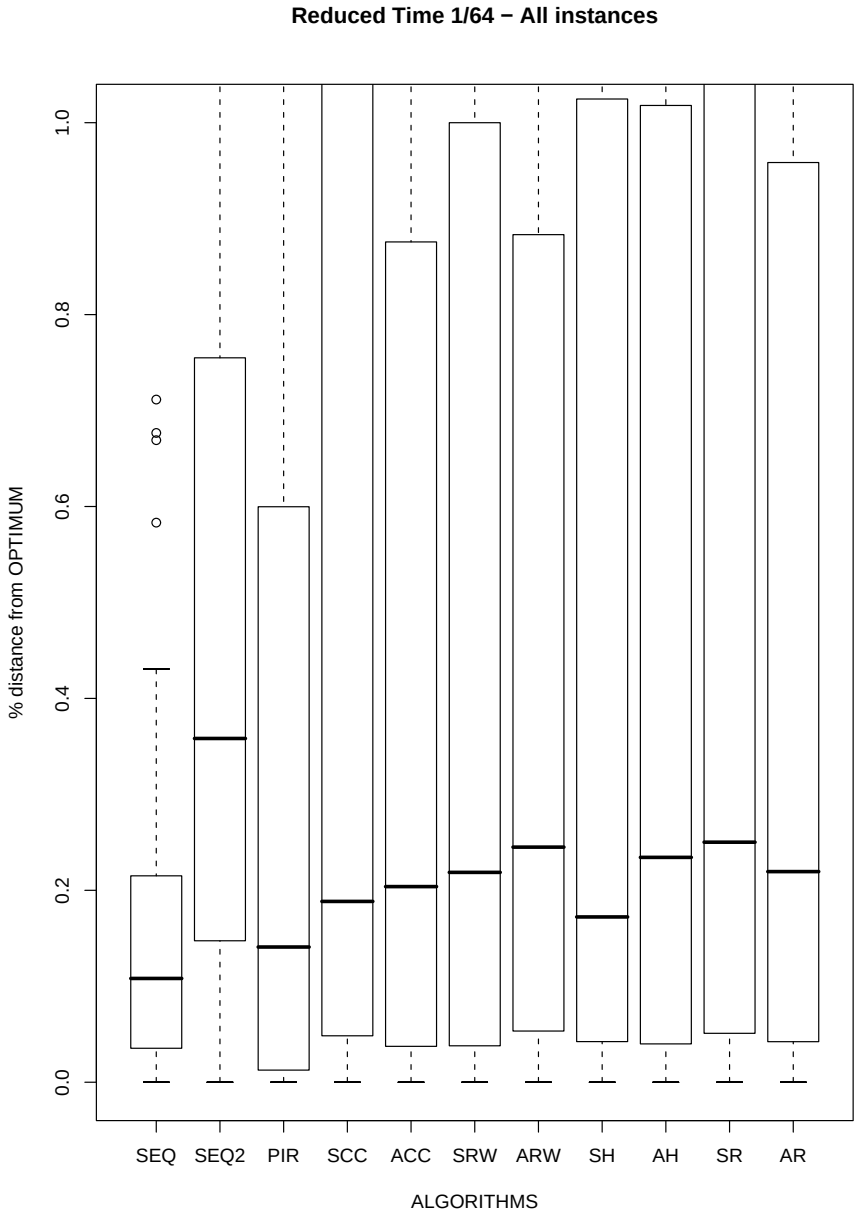


Fig. 7. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. Run time reduced by factor $\frac{1}{64}$

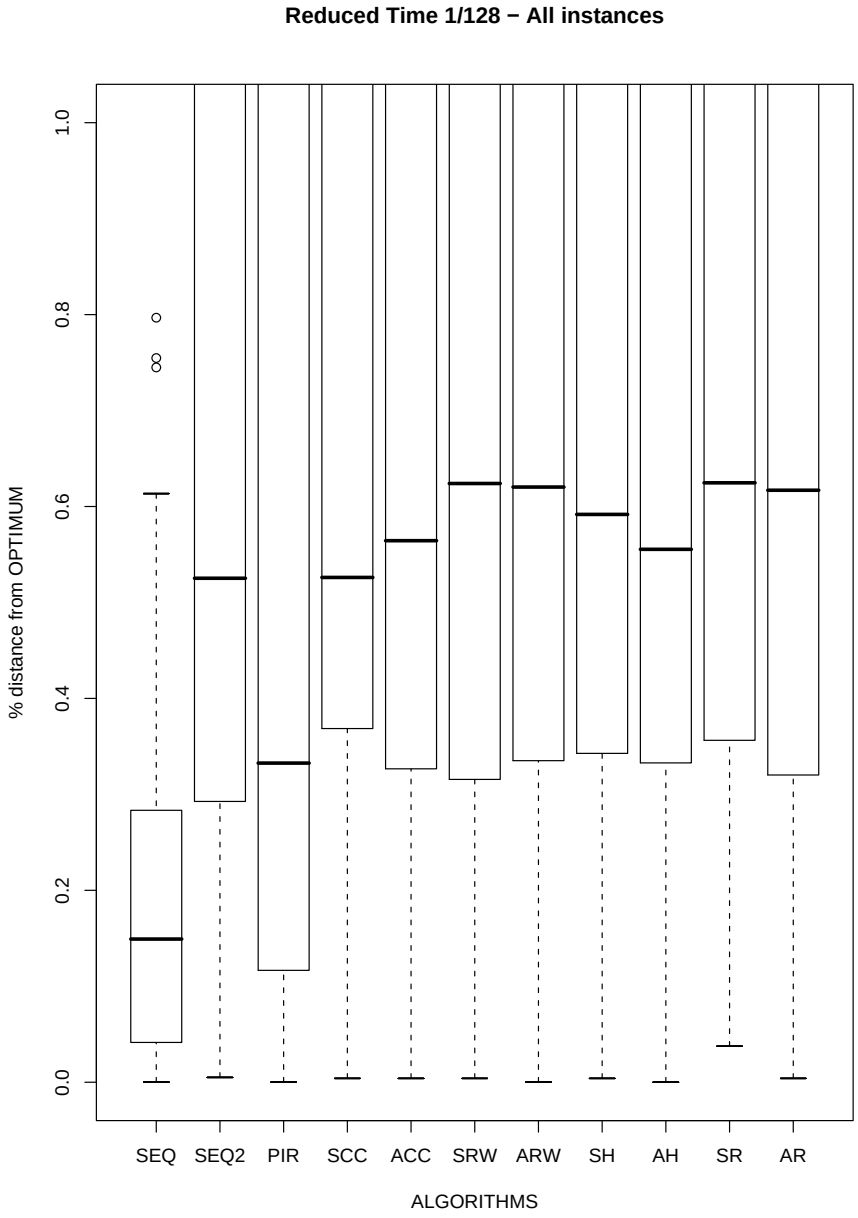
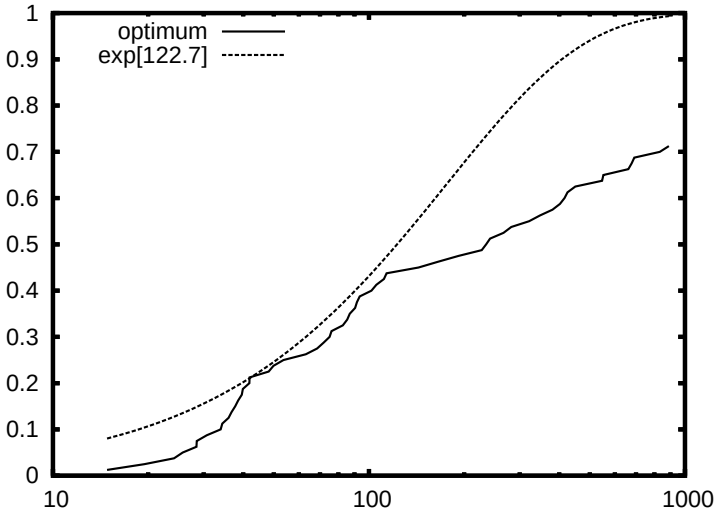
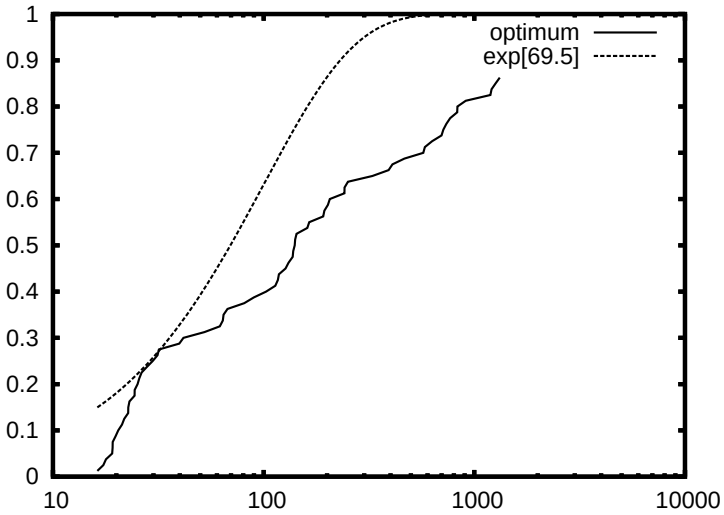


Fig. 8. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. Run time reduced by factor $\frac{1}{128}$



(a) pr1002



(b) d2103

Fig. 9. Run-time distribution over 80 independent trials of the sequential $\mathcal{MM}$AS algorithm for the instances pr1002 and d2103

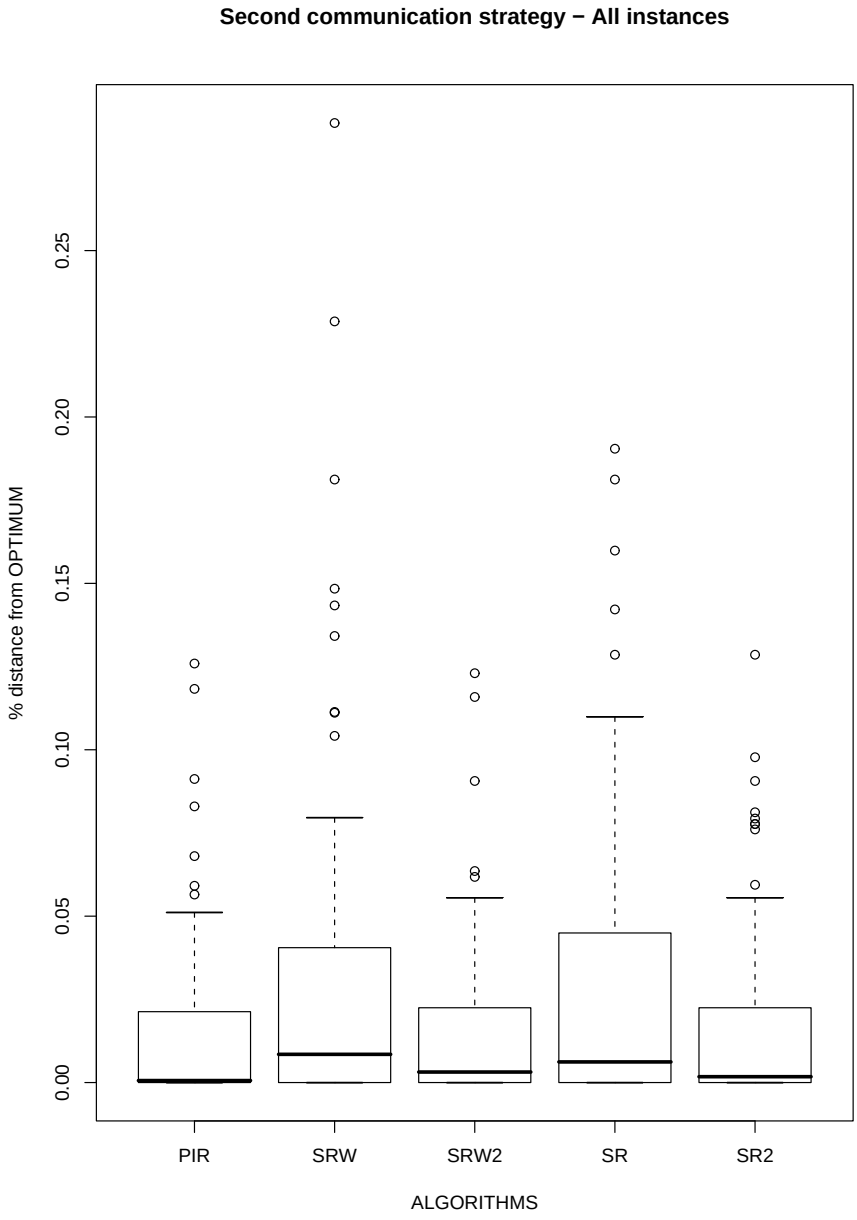


Fig. 10. Aggregate results over all instances. Boxplot of normalized results

Table 4. *p-values* for the null hypothesis "The distribution of the % distance from optimum of solutions for all instances is the same as PIR". The alternative hypothesis is that "The median of PIR distribution is lower". The significance level with which we reject the null hypothesis is 0.05

SRW	**SRW2**	**SR**	**SR2**
0.02	0.30	0.03	0.30

tions of the % distance from optimum of solutions for all instances is the same as PIR". The reduced frequency in communication has indeed a positive impact in the performance of the two parallel algorithms SRW2 and SR2, even though it is not sufficient to achieve better performance w.r.t the parallel independent runs. We believe that, to achieve better results than PIR we need to develop a more sophisticate communication scheme, that is dependent not only on the instance-size, but also on the run time.

5 Conclusions

The main contribution of this paper is the study of the impact on the performance produced by communication among multiple colonies interconnected with various topologies. We initially restricted the algorithms to the use of a constant communication rate among colonies to exchange the best-so-far solutions. For each topology, with the exception of the Parallel independent runs, we have developed two versions: a first one where the communication is synchronous, and a second one where the communication is asynchronous. We have shown that all the parallel models perform in average better than the equivalent sequential algorithms (SEQ and SEQ2).

The inability to restart of this version of $\mathcal{MM}$AS, transform the early convergence behavior into a stagnation behavior, biasing the performance in favor of PIR over all the other parallel models. We believe that better performance than PIR could be obtain by the parallel models either adding the restarting feature, or implementing communication schemes that avoid early convergence. This second approach could be achieved implementing the acceptance of solutions from other colonies only when they "differ" less than a certain number of components, leading to the creation of groups of colonies that search in different areas of the search space, or by exchanging the solutions with a frequency that depends on both instance size and run time.

Acknowledgments

This work is supported by 'COMP^{2}SYS' and by the 'ANTS' project. 'COMP^{2}SYS' is a Marie Curie Early Stage Training Site, funded by the European Communitys Sixth Framework Programme under contract number MEST-CT-2004-505079. The 'ANTS'

project is an Action de Recherche Concerte funded by the Scientific Research Directorate of the French Community of Belgium. M. Dorigo and T. Stützle acknowledge support from the Belgian National Fund for Scientific Research (FNRS), of which they are a Research Director and a Research Associate, respectively. 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 A. L. Christensen for his support during the MPI programming phase.

References

1. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J.: MPI: The Complete Reference - Volume 1: The MPI Core. Volume 1 of Scientific and Engineering Computation. The MIT Press (1998)
2. Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, E., Nitzberg, B., Saphir, W., Snir, M.: MPI: The Complete Reference - Volume 2: The MPI-2 Extensions. Volume 2 of Scientific and Engineering Computation. The MIT Press (1998)
3. Alba, E., ed.: Parallel Metaheuristics: A New Class of Algorithms. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, Hoboken, NJ (2005)
4. Stützle, T.: Parallelization strategies for ant colony optimization. In Eiben, A.E., et al., eds.: Parallel Problem Solving from Nature - PPSN V: 5th International Conference. Number 1498 in Lecture Notes in Computer Sciences, Germany, Springer-Verlag (1998) 722–731
5. Tanese, R.: Parallel genetic algorithms for a hypercube. In: Proceedings of the Second International Conference on Genetic algorithms and their application, Mahwah, NJ, Lawrence Erlbaum Associates, Inc. (1987) 177–183
6. Bullnheimer, B., Kotsis, G., Strauß, C.: Parallelization strategies for the Ant System. In Leone, R.D., et al., eds.: High Performance Algorithms and Software in Non-linear Optimization. Kluwer Academic Publishers (1998) 87–100
7. Talbi, E.G., Roux, O., Fonlupt, O., Robillard, D.: Parallel Ant Colonies for the Quadratic Assignment Problem. Future Generation Computer System **17**(4) (2001) 441–449
8. Middendorf, M., Reischle, F., Schmeck, H.: Multi colony ant algorithms. Journal of Heuristics **8**(3) (2002) 305–320
9. Piriyakumar, D.A.L., Levi, P.: A new approach to exploiting parallelism in ant colony optimization. In: International Symposium on Micromechatronics and Human Science (MHS) 2002, Nagoya, Japan. Proceedings, IEEE Xplore (2002) 237–243
10. Garey, M.R., Johnson, D.S.: Computers and Intractability / A Guide to the Theory of $\mathcal{NP}$-Completeness. W.H. Freeman & Company, San Francisco, CA (1979)
11. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy (1991)
12. Dorigo, M.: Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy (1992)
13. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)

14. Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: A critical survey. Annals of Operations Research **131** (2004) 373–395
15. Stützle, T., Hoos, H.H.: $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Future Generation Computer System **16**(8) (2000) 889–914
16. Grama, A., Gupta, A., Karypis, G., Kumar, V.: Introduction to parallel computing. Second edn. Pearson - Addison Wesley, Harlow, UK (2003)
17. Reinelt, G.: TSPLIB95 http://www.iwr.uni-heidelberg.de/groups/comopt/-software/tsplib95/index.html (2004)
18. Conover, W.J.: Practical Nonparametric Statistics. Third edn. John Wiley & Sons, New York, NY (1999)
19. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6** (1979) 65–70
20. Hoos, H., Sttzle, T.: Stochastic Local Search: Foundations & Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2004)