Parallel Ant Colony Optimization for the Traveling Salesman Problem

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

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{mmanfrin, mbiro, stuetzle, mdorigo}@iridia.ulb.ac.be

Abstract. There are two reasons for parallelizing a metaheuristic if one is 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 a 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). This makes at the time of this research clusters one of the most affordable and adopted parallel architectures for developing parallel algorithms.

The availability of parallel architectures at low cost has widened the interest for the parallelization of algorithms and metaheuristics [1]. When developing parallel population-based metaheuristics such as parallel genetic algorithms and parallel ant colony optimization (ACO) algorithms, it is common to adopt the "island model" approach [2], in which the exchange of information plays a major role. Solutions, pheromone matrices, and parameters have been tested (see for example [3,4,5,6]) as the object of such an exchange. In [3] solutions and pheromone levels are exchanged, producing a rather high volume of communication which requires a significant part of the computational time. In [6] the communication of the whole pheromone matrix leads to a decrease in solution quality as well as worse runtime behavior, while the exchange of best-so-far and elite solutions produces the best results w.r.t. solution quality. 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 multi-colony parallel ACO algorithm to solve the traveling salesman problem (TSP).

M. Dorigo et al. (Eds.): ANTS 2006, LNCS 4150, pp. 224–234, 2006.

We use the TSP, an $\mathcal{NP}$-hard problem [7], as a case study, that also has been a central test bed in the development of the ACO field. For each interconnection topology, we implement both synchronous and asynchronous communication. In the first case, the sender waits for the receiver to exchange messages. In the second case, the sender forwards the message and continues, not waiting for the receiver. The communication strategy we adopt involves the exchange of the best-so-far solutions every r iterations, after an initial period of "solitary" search. The main 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.

For this study, we use $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [8]—currently one of the best-performing ACO algorithms—as a basis for our parallel implementation. Our implementation of $\mathcal{MM}$AS is based on the publicly available ACOTSP code.[1]

Some research has been done on the parallelization of ACO algorithms, but, surprisingly enough, only few works used as a basis for the study of the effectiveness of the parallelization a high-performing ACO algorithm. An example is [9], where the effect of parallel independent runs was studied.

The article is structured as follows. Section 2 describes the details of our implementation of $\mathcal{MM}$AS, 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

ACO is a metaheuristic introduced in 1991 by Dorigo and co-workers [10,11]. For an overview of the currently available ACO algorithms see [12]. 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 [13].

As said in Section 1, for this research, we use $\mathcal{MM}$AS as a basis for our parallel implementation. We extended the ACOTSP code by quadrant nearest neighbor lists. To have a version that is easily parallelizable, we removed the occasional pheromone re-initializations applied in the $\mathcal{MM}$AS described in [8], and we use only a best-so-far pheromone update. Our version also uses the 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

[1] `http://www.aco-metaheuristic.org/aco-code/public-software.html`

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. The topologies we studied are:

Fully-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. We consider a synchronous and an asynchronous implementation of this model identified by SFC and AFC, respectively, in the following.

Replace-worst. This parallel model is similar to the fully-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 sends only one message to the best colony, containing the identity of the worst colony, and the best colony sends its best-so-far solution only to the worst colony. We consider a synchronous and an asynchronous implementation of this model identified by SRW and ARW, respectively, in the following.

Hypercube. In this model, k colonies are connected according to the hypercube topology (see [14] for a detailed explanation of this topology). Practically, each colony is located on a vertex i of the hypercube and can communicate only with the 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. We consider a synchronous and an asynchronous implementation of this model respectively SH and AH in the following.

Ring. Here, k colonies are connected in such a way that they create a ring. 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 only the best-so-far solution from colony $[(i-1+k) \bmod k]$. We consider a synchronous and an asynchronous implementation of this model, called SR and AR in the following.

Parallel independent runs. In this 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 k runs. Using parallel independent runs is appealing as basically no communication overhead is involved and nearly no additional implementation effort is necessary. In the following, 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 fully-connected, to more local communication, as in ring, to basically no communication, as in parallel independent runs.

3 Experimental Setup

As said in Section 1, all algorithms are based on the ACOTSP software, which is coded in C. The parallel algorithms add, w.r.t. the sequential code, the communication capability, using MPI libraries. 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^{TM} 244 CPUs, 2 GB of RAM, and one 1 Gbit Ethernet network card. The nodes are interconnected through a 48-ports Gbit switch.

The initial 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 TSPLIB [15] 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. We considered two cases: in the first one (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 one (SEQ2), it runs for the same wall-clock time as one CPU of the parallel algorithm. It is reasonable to request that a parallel algorithm performs at least not worse than SEQ2 within the computation time under consideration.

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 [8], 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

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

optimal solution; the normalized value is then defined as

$$\text{Normalized Value for MH} = \frac{c_{\text{MH}} - c_{\text{opt}}}{c_{\text{opt}}} \cdot 100. \quad (1)$$

This normalization method provides a measure of performance that is independent of the values of the different optimal solutions, allowing us to aggregate results form different instances.

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 for high quality solutions, having a positive impact on the performance of the algorithms. In order to test the effects of communication, we implement versions of $\mathcal{MM}$AS algorithm that differ only in the communication behavior. 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[2] grouped by algorithm over all instances after the normalization described in Section 3. The boxplot indicates that, on average, all the parallel models, except SFC, 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 [16] with *p-values* adjusted by Holm's method [17].

As can be seen from Table 2, the differences in performance of all the parallel models with information exchange from those of PIR are statistically significant; this confirms 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 interconnection topology, using again the Wilcoxon test with p-values adjusted by Holm's method and we report the results in Table 3. Differences in performance among interconnection topologies are not statistically significant.

Even though the boxplot indicates that parallel algorithms achieve, on average, better performance than the sequential ones, the impact of communication on performance seems negative. One reason might be that the run times are rather high, and $\mathcal{MM}$AS easily converges in those times. PIR can count on multiple independent search paths to explore the search space, reducing the effects of the "stagnation" behavior. In fact, the other parallel algorithms accelerate the convergence toward a same solution, due to the frequent exchange of information, as can be verified by the traces of the algorithms' outputs.

[2] We refer the reader interested in the raw data to the URL: `http://iridia.ulb.ac.be/supp/IridiaSupp2006-001/`

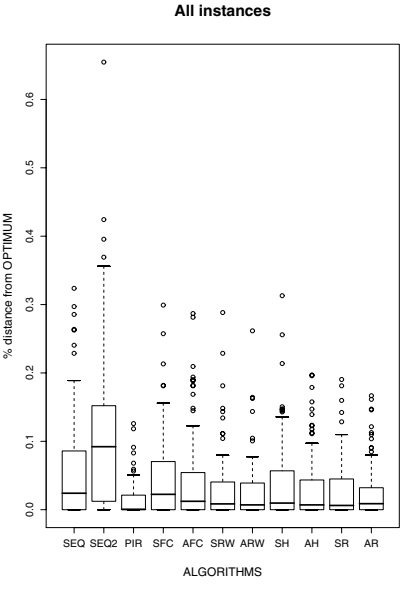


Fig. 1. Aggregate results over all instances. Boxplot of normalized results

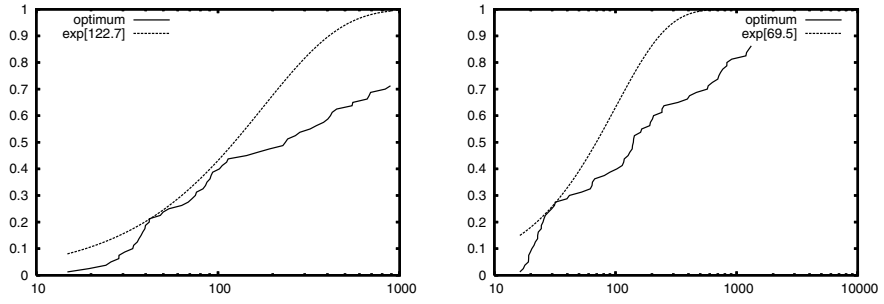
Table 2. *p-values* for the null hypothesis "The distribution of the % distance from optimum of solutions for all instances is the same as PIR". The alternative hypothesis is that "The median of the PIR distribution is lower". The significance level with which we reject the null hypothesis is 0.05.

SFC	**AFC**	**SRW**	**ARW**	**SH**	**AH**	**SR**	**AR**
5.4e-4	0.01	0.02	0.02	1.2e-3	0.02	0.02	0.02

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

	FC	**RW**	**H**
RW	0.55	-	-
H	1	1	-
R	0.55	1	1

Run time distributions. To examine the possibility of the "stagnation" behavior, we analyze the run-time distribution (RTD) of the sequential algorithm. A qualified run-time distribution measures the distribution of the time a stochastic local search algorithm requires to reach a specific target of solution quality, for example the optimal solution value. In Figure 2 we give plots of the measured RTDs for reaching the known optimal solution value for the two instances pr1002 and d2103. As explained in [18], the exponential distribution that is given



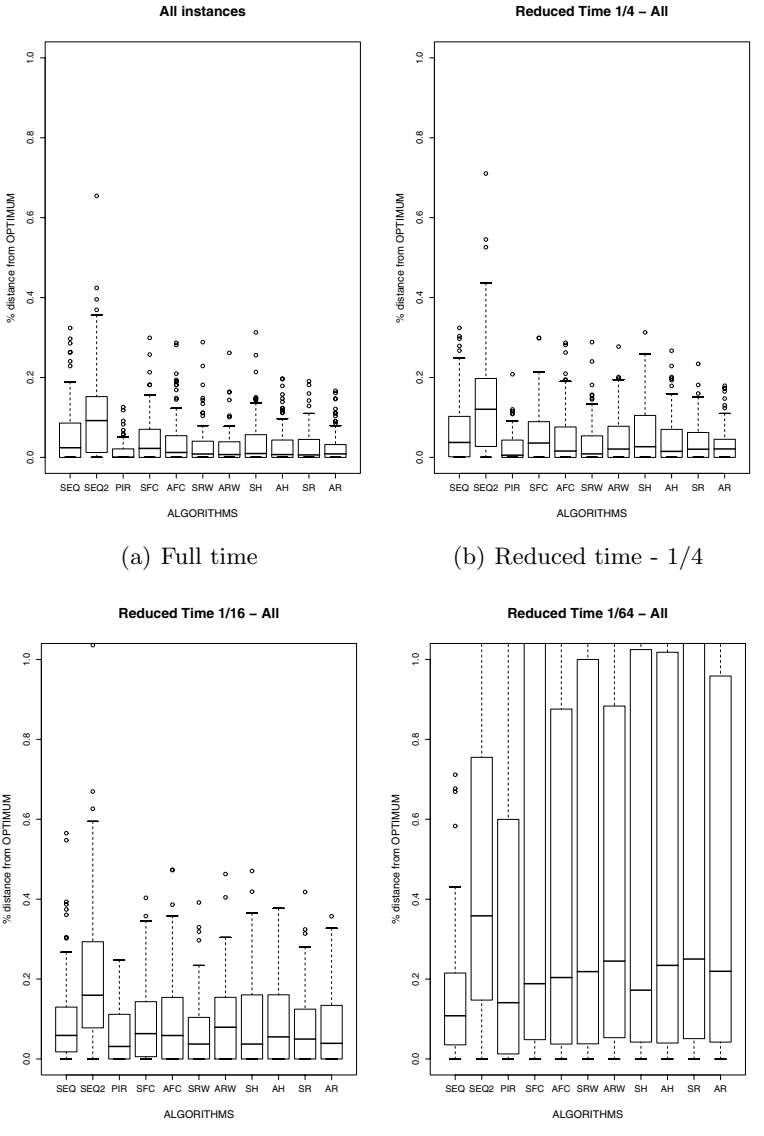
(a) pr1002 (b) d2103

Fig. 2. Run-time distribution over 80 independent trials of the sequential $\mathcal{MM}$AS algorithm for the instances pr1002 and d2103

in these plots indicates that this version of $\mathcal{MM}$AS may profit from algorithm restarts (essentially, restarting after an appropriately chosen cutoff time, one can force the empirical RTD to follow the exponential distribution due to its statistical properties) and, hence, this is an indication of stagnation behavior. This explains to a large extent the good perfomance of parallel independent run given that PIR can count on multiple independent search paths to explore the search space, reducing the effects of the stagnation behavior.

Performance for reduced run-times. In order to check if our doubt on the "stagnation" behavior has some fundament, we re-analyze the results considering run times that are 1/4, 1/16, and 1/64 of the values reported in Table 1, showing the resulting boxplots in Figure 3. We observe that the more we reduce the run time, the smaller are the differences between the performance of the SEQ algorithm and the others, up to the reduced time of 1/64, for which SEQ performs on average better than all the parallel models (remember that SEQ has a run time that is 8-times the wall-clock time of a parallel algorithm).

Frequency of communication. As indicated, an apparent problem of our communication scheme is that communication is too frequent. To better understand the impact that the frequency of communication has on performance, we change the communication scheme to an exchange every $n/4$ iterations, except during the first $n/2$, where n is the size of the instance. We test this new communication scheme on the parallel models replace-worst (SRW2) and ring (SR2). Figure 4 shows the boxplots of the results. Once more, we rely on the Wilcoxon test with Holm's adjustment to verify whether 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 distributions of the % distance from optimum of solutions for all instances is the same as PIR" with a significance level of 0.05, given that the *p-values* relative to SRW2 and SR2 are both equal to 0.30. The reduced



(a) Full time

(b) Reduced time - 1/4

(c) Reduced time - 1/16

(d) Reduced time - 1/64

Fig. 3. Aggregate results over all instances. Boxplots of normalized results restricted to values in [0,1]. (a) full time is used; (b) run time reduced to 1/4 of full time; (c) run time reduced to 1/16 of full time; (d) run time reduced to 1/64 of full time.

frequency in communication has indeed a positive impact on the performance of the two parallel algorithms SRW2 and SR2, even though this is not sufficient to achieve better performance w.r.t parallel independent runs. We strongly believe

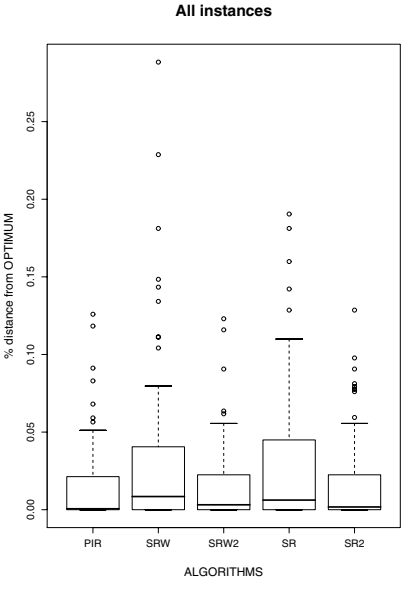


Fig. 4. Aggregate results over all instances. Boxplot of normalized results.

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 of communication among multiple colonies interconnected with various topologies on the final solution quality reached. 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 in which the communication is synchronous, and a second one in which the communication is asynchronous. We have shown that all the parallel models perform on average better than the equivalent sequential algorithms (SEQ and SEQ2).

As stated in Section 2, to have a version that was easy to parallelize, we removed from the $\mathcal{MM}$AS implementation the occasional pheromone re-initialization and we used only a best-so-far pheromone update. These modifications result in a stagnation behavior of the sequential algorithm; this stagnation behavior can be avoided to a large extent by parallel independent runs, which also explains its overall good behavior, biasing the performance in favor of PIR over all the other parallel models. We believe that better performance than PIR can be obtained 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 Community's Sixth Framework Programme under contract number MEST-CT-2004-505079. The ANTS project is an Action de Recherche Concertée 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. Alba, E., ed.: Parallel Metaheuristics: A New Class of Algorithms. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, Hoboken, NJ (2005)
2. Tanese, R.: Parallel genetic algorithms for a hypercube. In: Proceedings of the second international conference on Genetic Algorithms and their Applications, Hillsdale, NJ, Lawrence Erlbaum Associates, Inc. (1987) 177–183
3. Bullnheimer, B., Kotsis, G., Strauß, C.: Parallelization strategies for the Ant System. In De Leone, R., et al., eds.: High Performance Algorithms and Software in Nonlinear Optimization. Kluwer Academic Publishers, Norwell, MA (1998) 87–100
4. Middendorf, M., Reischle, F., Schmeck, H.: Multi colony ant algorithms. Journal of Heuristics **8**(3) (2002) 305–320
5. 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 Standard Office (2002) 237–243
6. Benkner, S., Doerner, K.F., Hartl, R.F., Kiechle, G., Lucka, M.: Communication strategies for parallel cooperative ant colony optimization on clusters and grids. In: Complimentary Proceedings of PARA'04 Workshop on State-of-the-Art in Scientific Computing, June 20-23, 2004, Lyngby, Denmark (2005) 3–12
7. 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)
8. Stützle, T., Hoos, H.H.: $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. Future Generation Computer System **16**(8) (2000) 889–914
9. Stützle, T.: Parallelization strategies for ant colony optimization. In Eiben, A.E., et al., eds.: Parallel Problem Solving from Nature - PPSN V. Volume 1498 of Lecture Notes in Computer Sciences., Berlin, Germany, Springer-Verlag, (1998) 722–731

10. 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)
11. Dorigo, M.: Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy (1992)
12. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)
13. 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
14. Grama, A., Gupta, A., Karypis, G., Kumar, V.: Introduction to parallel computing. Second edn. Pearson - Addison Wesley, Harlow, UK (2003)
15. Reinelt, G.: TSPLIB95 `http://www.iwr.uni-heidelberg.de/groups/comopt/-software/tsplib95/index.html` (2004)
16. Conover, W.J.: Practical Nonparametric Statistics. Third edn. John Wiley & Sons, New York, NY (1999)
17. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics **6** (1979) 65–70
18. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations & Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA (2004)