An Experimental Investigation of Iterated Ants for the Quadratic Assignment Problem

Wolfram Wiesemann and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2006-003

February 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/2006-003

An Experimental Investigation of Iterated Ants for the Quadratic Assignment Problem

Wolfram Wiesemann
Fachbereich Wirtschaftswissenschaften
Technische Universität Darmstadt
Darmstadt, Germany
E-mail: wolfram.wiesemann@google.com

Thomas Stützle
IRIDIA, CP194/6
Université Libre de Bruxelles
1050 Brussels, Belgium
E-mail: stuetzle@ulb.ac.be

Abstract

Ant Colony Optimization (ACO) algorithms construct solutions each time starting from scratch, that is, from an *empty* solution. Similar to ACO, Iterated Greedy is a constructive stochastic local search (SLS) method. However, differently from ACO, Iterated Greedy starts the solution construction from partial solutions. In this paper we examine the performance of a variation of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, one of the most successful ACO algorithms, that exploits this idea of starting the solution construction from partial solutions when applied to the quadratic assignment problem. This particular application problem is chosen, since another source of inspiration for this article are earlier researches on the usage of *external memory* in ACO, an idea that actually also results in the usage of partial solutions to seed the solution construction in ACO algorithms. Differently from previously reported results on external memory usage, our computational results are more pessimistic in the sense that the idea of using partial solutions for seeding the constructions does not necessarily lead to improvements, when this idea is integrated into very high-performing ACO algorithms.

1 Introduction

Ant Colony Optimization (ACO) algorithms generate candidate solutions for an optimization problem by a construction mechanism where the choice of the solution component to be added at each construction step is done probabilistically biased by (artificial) pheromone trails and heuristic information on the problem under consideration [6]. In usual ACO algorithms, each (artificial) ant starts from an initially empty solution and adds solution components to its current partial solution until a complete candidate solution is obtained.

In this article we examine the possibility of changing this feature underlying most ACO algorithms by starting the solution construction from partial solutions that are obtained by removing some solution components of an ant's current solution. This modification of ACO algorithms is in large part inspired by the Iterated Greedy (IG) method. IG algorithms iterate over construction algorithms in the following way. Given some initial solution s, first some solution components are removed, resulting in a partial candidate solution s_p. Starting from s_p a complete candidate solution s' is reconstructed by a greedy construction heuristic. An acceptance criterion then decides from which of the two solutions s and s' the next iteration continues. Iterated Greedy algorithms are at the core of high-performing algorithms for the set covering problem [3, 9, 11] and are state-of-the-art for the permutation flow-shop problem [12]. The idea underlying IG can be transferred in a straightforward way to ACO algorithms. The extension followed here essentially says that some solution components of an ant's current solution are removed according to some rule and complete candidate solutions are reconstructed, following the usual construction rules of ACO algorithms. This process is here integrated into the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) algorithm [16], resulting into an iterated ants $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (ia$\mathcal{MM}$AS).

Another line of research that inspired this paper is the usage of external memory in ACO algorithms as proposed in the papers of Acan [1, 2]. There, external memory involves the storage of partial solutions extracted from the best solutions obtained after each iteration of the ACO algorithm. Essentially, the partial solutions comprise a number of randomly chosen solution components of complete solutions. From these partial solutions, an ant chooses one according to a selection scheme that gives preference to better partial solutions. This idea was tested by Acan using a modified $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System algorithm for the Travelling Salesman Problem (TSP) [1] and the Quadratic Assignment Problem (QAP) [1, 2] as a case study. Although in these papers the "iterated ants" variant of $\mathcal{MM}$AS was shown to reach better average solution qualities than a re-implementation of $\mathcal{MM}$AS, the overall computational results were rather poor (in fact, by far worse than previously published results with $\mathcal{MM}$AS, as can easily be seen when comparing the results across the papers [16] and [1, 2]), possibly because no reasonably well performing local search was included to improve the ants' solutions. Differently, in this article we extend a high-performing ACO algorithm by the iterated ants, namely the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System algorithm for the QAP by Stützle and Hoos [16], which is known to be a top-class performing ACO algorithm for the QAP [15]. This is done, because we think it to be essential to test the usefulness of adding new features to ACO algorithms by adding these to high-performing algorithms, since only then it can be judged whether the new ideas can contribute to the state-of-the-art in a particular field.

The remainder of this article is structured as follows. In Section 2 we give the most important details of the $\mathcal{MM}$AS application for the QAP and the extension we used to implement the ia$\mathcal{MM}$AS algorithm. Section 3 gives the results of an extensive computational study and we conclude in Section 4.

2 $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and iterated ants for the QAP

Quadratic assignment problem. The quadratic assignment problem (QAP) is a widely studied $\mathcal{NP}$-hard optimization problem [13] that is an abstract model of many real-life problems arising in the location of facilities like units in a hospital or the layout of keyboards [4, 5]. In the QAP, one is given two $n \times n$ matrices A and B, where a_{ij} is the distance between locations i and j and b_{kl} is the flow (for example, of material, patients etc.) between units k and l. The cost contribution of assigning units k and l to locations i and j, respectively, amounts to $a_{ij} \cdot b_{kl}$ and the goal is to minimize the cost arising of all interactions. A solution for the QAP can be represented by a permutation π, where $\pi(i)$ gives the unit assigned to location i. The objective function of the QAP is then

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot b_{\pi(i)\pi(j)}. \tag{1}$$

The QAP is one of the hardest optimization problems to solve to optimality. Currently, SLS algorithms define the state-of-the-art approaches for finding near-optimal solutions in a reasonable amount of time [8]. Among the various available SLS methods, ACO has been shown to be particularly successful on the QAP [15] and among the various existing ACO algorithms, $\mathcal{MM}$AS was shown to be among the top performers [14, 16].

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the QAP. In the $\mathcal{MM}$AS algorithm for the QAP ($\mathcal{MM}$AS-QAP), the artificial pheromone trails τ_{ij} give the desirability of assigning a unit j to location i. The solution construction of an ant in $\mathcal{MM}$AS-QAP is done by first ordering the locations randomly and then assigning in the logical construction step l a unit to the location at the lth position. The probability of assigning a still unassigned unit j to a location i in this step is given by

$$p_{ij} = \frac{\tau_{ij}}{\sum_{l \in \mathcal{N}(i)} \tau_{il}}, \tag{2}$$

where $\mathcal{N}(i)$ is the set of still unassigned units, that is, those units that still are to be assigned to some location. Two remarks are noteworthy here. First, $\mathcal{MM}$AS-QAP does not use any heuristic information in the solution construction. Second, two variants of $\mathcal{MM}$AS-QAP were proposed. A first makes at each construction step a probabilistic choice according to Equation 2 [16]; a second one uses instead the pseudo-random proportional action choice rule originally proposed for Ant Colony System [14, 18]. Here, we focus on the first version. The pheromone update in $\mathcal{MM}$AS-QAP is done by lowering the pheromone trails by a constant factor ρ and depositing pheromone on the individual assignments of either the best solution in the current iteration, the best solution found so far by the algorithm or the best solution found since the last

```
procedure Iterated_Greedy
    $s_0 := GenerateInitialSolution$
    $s := LocalSearch(s_0)$
    repeat
        $s_p := Destruct(s)$
        $s' := Construct(s_p)$
        $s' := LocalSearch(s')$        % optional
        $s := AcceptanceCriterion(s, s')$
    until termination condition met
end
```

Figure 1: Outline of an IG algorithm. s_p is a partial candidate solution.

re-initialization of the pheromone trails. A pheromone reinitrialization is triggered if the branching factor is below a certain threshold and for a given number of iterations no improved candidate solution has been found. For details on $\mathcal{MM}$AS-QAP and its parameter settings we refer to [16] (the only difference to the earlier proposed parameter settings was the usage of a setting of $\rho = 0.1$ instead of the earlier $\rho = 0.8$ because of a slightly improved performance.

$\mathcal{MM}$AS-QAP uses local search for improving each candidate solution generated by the ants. Here, we use an iterative improvement algorithm in the 2–exchange neighborhood, where two candidate solutions are neighbored if they differ in the assignment of exactly 2 units to locations. The local search algorithm uses a best-improvement pivoting rule.

Iterated ants. In ACO algorithms, candidate solutions are generated by starting each solution construction from scratch. However, other constructive SLS methods are known that use repeated solution constructions as well but start from partial solutions. This is the central idea of *Iterated Greedy* (IG). IG algorithms start with some complete candidate solution and then cycle through a main loop consisting of three procedures; first, a procedure *Destruct* removes from a complete solution s a number of solution components, resulting in a partial candidate solution s_p. Starting from s_p, next a complete candidate solution s' is reconstructed by a procedure *Construct* and finally, a procedure *AcceptanceCriterion* decides whether the next iteration is continued from s or s'. In addition, it is straightforward to also include a local search procedure that improves a candidate solution, once it is completed by *Construct*. This results in the overall outline of an IG algorithm that is given in Figure 1.

Iterated ants applies the concept of IG to ACO. This can be done in a rather straightforward way by considering each ant as implementing an IG algorithm. That is, an individual ant follows the same steps of an IG algorithm and, hence, the solution construction in the ACO-IG hybrid algorithm starts from a partial solution that is obtained from deleting solution components from a complete candidate solution of an ant. The solution construction by the ants follows the same steps as usual in ACO algorithms, that is, solution components are added taking into account pheromones and possibly heuristic information.

In this article, we integrated this idea directly into the $\mathcal{MM}$AS algorithm and tested it for the QAP. We study the behavior of the algorithm considering various choices for the procedures *Destruct*, where we varied the choice of how solution components are removed from complete solutions, how many solution components are removed, and the type of pheromone update chosen in the algorithm. (Note that removing a solution component for the QAP refers to undoing an assignment of a unit to a location). For the solution components to be removed, we study three variants.

- `rand`: the solution components to be removed are chosen randomly according to a uniform distribution.
- `prob`: the probability of removing a solution component is proportional to the corresponding pheromone trail τ_{ij}, that is, the higher the pheromone trail, the more likely it is to remove a solution component.
- `iprob`: the probability of removing a solution component is inversely proportional to the associated pheromone trail τ_{ij}, that is, the lower the associated pheromone trail, the more likely it is to remove a solution component.

	tai60a	**tai80a**	**sko81**	**kra30a**	**ste36a**	**tai60b**	**tai80b**	**tai100b**
`var-rand-gb`$^+$	**1.93**	**1.50**	**0.23**	0.36	**0.21**	0.23	0.66	0.38
`var-rand-gb`$^-$	2.03	1.54	0.23	**0.21**	0.27	**0.07**	**0.48**	**0.32**
p-value	0.2810	0.9062	0.4576	0.9938	0.9062	0.4676	0.1545	0.6994

Table 1: Comparison of variant `var-rand-gb`$^+$ and `var-rand-gb`$^-$. The best results for each instance are indicated in bold-face.

Regarding the number of solution components to be removed we consider two different possibilities.

- `fixed(l)`: Of each ant exactly l solution components are removed, where l is a parameter. In the experimental study, various values for l were tested.
- `variable`: In this case, the number of solution components to be removed is not fixed *a priori*, but is maintained variable similar to how the perturbation strength is modified in the simple variable neighborhood search: if for a current value l no improved solution is obtained for the ant, we set $l := l + 1$: otherwise, l is set to some minimum value.

Finally, we also consider two possibilities for the pheromone update rule in $\mathcal{MM}$AS-QAP.

- `gb`$^+$: The pheromone update rule is the very same as the one used in the original $\mathcal{MM}$AS-QAP algorithm.
- `gb`$^-$: The best-so-far candidate solution and the best candidate solution since the pheromone re-initialization are not taken into account when depositing pheromone, that is, only the iteration-best ant deposits pheromone.

3 Computational study

We tested the iterated ants for the QAP on eight instances ranging in size from $n = 30$ to $n = 100$ from QAPLIB [7]. The eight instances are chosen such that they have different instance characteristics, representative for most of the available QAP instances that can be found at QAPLIB. The tested instances include (i) ones, where both matrix entries are generated according uniform distribution, (ii) instances, where the distance matrix corresponds to Manhattan distances of locations on a grid, (iii) some instances derived from real-life applications of the QAP, and (iv) randomly generated instances that resemble the structure of real-life QAP instances.

All the experimental results, except where indicated differently, are measured across 25 independent trials of the algorithms and the code was run on a IBM X31 ThinkPad-Notebook with a Pentium M 1.5 GHz processor, 512 MB RAM and 1024 kB L2-Cache. For all instances we first run the reference strategy, $\mathcal{MM}$AS-QAP for 500 iterations and measure the average time to finish the trials. This stopping time is then taken as the termination criterion for all variants; that is, all variants are given the same computation time. We compare the algorithms using the average percentage excess over the best-known solutions (proven optimal solutions are only available for two instances `kra30a` and `ste36a`) and for each comparison we use the non-parametric Wilcoxon test to check the statistical significance of the observed differences in performance.

3.1 Study of ia$\mathcal{MM}$AS parameters

In this section, we present the results on the parameter settings for the ia$\mathcal{MM}$AS-QAP algorithm, while in the next one, we present an analysis of the run-time behavior of ia$\mathcal{MM}$AS-QAP and a comparison with $\mathcal{MM}$AS-QAP. In the following we refer to the variants by using abbreviations in dependence of the choice for the three stategies; for example, `var-rand-gb`$^+$ refers to the variant that uses a variable number of components to delete, the components are chosen randomly for deletion, and the usual $\mathcal{MM}$AS pheromone update rule is chosen using the best-so-far solution in the update.

	tai60a	tai80a	sko81	kra30a	ste36a	tai60b	tai80b	tai100b
`var-prob-gb`$^+$	**1.99**	1.60	0.27	0.25	**0.33**	0.26	0.79	0.35
`var-prob-gb`$^-$	2.04	**1.58**	**0.26**	**0.19**	0.48	**0.11**	**0.54**	**0.31**
p-value	0.4676	0.9062	0.2810	0.9938	0.2810	0.2810	0.1545	0.1545

Table 2: Comparison of variant `var-prob-gb`$^+$ and `var-prob-gb`$^-$. The best results for each instance are indicated in bold-face.

	tai60a	tai80a	sko81	kra30a	ste36a	tai60b	tai80b	tai100b
`var-iprob-gb`$^+$	**1.77**	**1.38**	**0.18**	0.45	**0.12**	**0.17**	0.52	0.32
`var-iprob-gb`$^-$	2.00	1.51	0.20	**0.37**	0.17	0.18	**0.49**	**0.28**
p-value	**0.0002**	**0.0148**	0.4676	1.0000	0.9062	0.9938	0.6994	0.9062

Table 3: Comparison of variant `var-iprob-gb`$^+$ and `var-iprob-gb`$^-$. The best results for each instance are indicated in bold-face.

Pheromone update. First, we tested the two possible variants for the pheromone update, namely `gb`$^+$ and `gb`$^-$. In this test, we used only the `variable` strategy for the number of solution components to be removed and we considered all three possible choices of {`rand`, `prob`, `iprob`}. As can be seen from the results that are given in Tables 1 to 3, the differences between the two strategies `gb`$^+$ and `gb`$^-$ are, independent of the way solution components are removed, rather minor and and most of the observed differences are statistically insignificant. Since for the few cases, where statistically significant differences were observed, these were in favor of the strategy `gb`$^+$, we continue to use this one for the remainder of the paper. (Note that additional tests confirmed that this conclusion is the same if the `fixed(l)` strategy were chosen.)

Strategies for choosing solution components for removals. As a next step, we examined the influence of the type of deletions of solution components, which is one of {`rand`, `prob`, `iprob`}. The computational results, given in Table 4, indicate that the strategy `iprob` gives best overall results. On many instances the performance of `iprob` is statistically better than that of `prob` and for all instances the average solution quality obtained by `iprob` is better than that of `rand`. Note that `iprob` corresponds to an intensification of the search compared to the other two variants, since assignments that have associated a high pheromone value are more likely to remain in partial solutions.

Number of removals. In a final step, we examined the influence of the two strategies for removing solution components (`variable` vs. `fixed(l)`) and for the latter also the settings for the parameter l that determines how many solution components are removed. (The results of `var-iprob-gb`$^+$ can be taken from Table 4.) Ideally, we would like to see a pattern that suggests good settings for l. Yet, an inspection of Table 5 shows that there is no clear trend observable (best average performance is in boldface). One may tentatively conjecture that except for two instances with uniformly random distance and flow matrices (`tai60a` and `tai80a`) the number of solution components to be removed can be large, to be on the safe side (note that for four of the instances the lowest average deviation from the best known solutions is reached for the highest value tested for the parameters).

	tai60a	tai80a	sko81	kra30a	ste36a	tai60b	tai80b	tai100b
`var-rand-gb`$^+$ (1)	1.93	1.50	0.23	0.36	0.21	0.23	0.66	0.38
`var-prob-gb`$^+$ (2)	1.99	1.60	0.27	**0.25**	0.33	0.26	0.79	0.35
`var-iprob-gb`$^+$ (3)	**1.77**	**1.38**	**0.18**	0.45	**0.12**	**0.17**	**0.52**	**0.32**
p-value (1)/(2)	0.4676	0.1545	0.0366	0.9062	0.2810	**0.0158**	0.2810	0.2850
p-value (1)/(3)	**0.0063**	**0.0056**	0.2810	0.6994	0.2810	0.9062	0.1545	0.9938
p-value (2)/(3)	**0.0002**	$\approx$ **0**	**0.0002**	0.6994	0.0783	**0.0063**	**0.0023**	0.9062

Table 4: Comparison of the variants `var-rand-gb`$^+$, `var-prob-gb`$^+$ and `var-iprob-gb`$^+$. We indicate in boldface the p-values for which the corresponding hypothesis test on equal performance would be rejected after α-corrections according to Bonferroni due to the multiple comparisons.

	tai60a	tai80a	sko81	kra30a	ste36a	tai60b	tai80b	tai100b
k = 10	1.59	**1.24**	0.17	0.79	0.24	0.27	0.86	0.61
k = 20	1.68	1.38	0.11	**0.22**	0.31	0.11	**0.25**	0.11
k = 30	**1.43**	1.57	0.07	–	**0.06**	0.17	0.42	0.10
k = 40	1.53	1.37	0.11	–	–	0.25	0.58	0.16
k = 50	1.69	1.36	0.10	–	–	**0.00**	0.47	0.10
k = 60	–	1.32	**0.01**	–	–	–	0.31	0.10
k = 70	–	1.28	0.12	–	–	–	0.38	**0.07**

Table 5: Comparison of various values of the parameter k in variant `fixed(k)-iprob-gb`$^+$. The best results for each instance are indicated in bold-face.

	tai60a	tai80a	sko81	kra30a	ste36a	tai60b	tai80b	tai100b
$q_{0.25}$	8.57	20.22	25.81	1.30	2.35	10.87	26.15	51.12
$q_{0.5}$	8.61	20.35	25.89	1.31	2.37	10.97	26.41	52.19
$q_{0.75}$	8.64	20.49	26.08	1.31	2.37	11.00	26.87	53.05

Table 6: Execution time for trials of 500 iterations for the reference strategy on the tested instances. Given are the 0.25, 0.5, and 0.75 quantiles of the measured times.

3.2 Comparison of $\mathcal{MM}$AS and ia$\mathcal{MM}$AS

As a next step, we compare the performance of the original $\mathcal{MM}$AS-QAP with ia$\mathcal{MM}$AS-QAP. Before comparing their performance in terms of solution quality reached after a same computation time, we first consider the difference in the number of iterations applicable in a same computation time for the variants.

Speed. Table 6 gives the computation times $\mathcal{MM}$AS-QAP requires to complete 500 iterations, while Table 7 summarizes the number of iterations the three variants can apply until meeting the termination criterion (average time taken by $\mathcal{MM}$AS-QAP). As can be seen, the `variable` strategy and the setting `k=30` for most instances can do many more iterations than $\mathcal{MM}$AS-QAP in a same computation time. This is part is due to the more rapid solution construction. However, in the QAP the interpretation of this increased number of iterations must be slightly different, because when profiling the code, one notices that about 98% of the computation time are due to the local search. Hence, an explanation for the increased number of iterations rather is that the solutions generated by the iterated ants require less iterations of the iterative improvement local search, probably because a part of the solution is maintained from a previous local optimum. (Recall that in our algorithm we make the usual usage of local search in ACO and that removals of solution components start from locally optimal solutions.)

Comparison based on summary statistics. In a next step, we compare the performance of the reference strategy, $\mathcal{MM}$AS-QAP to the best performing variant using the `variable` strategy and two parameter settings for the `fixed(l)` strategy–one keeping $l = 30$ constant and one choosing for each instance the settings that gave the best average solution quality. The comparison to the latter case is certainly unfair; however, it is interesting because it gives an impression of what would be the best case performance if for each instance we would know the appropriate parameter setting for the `fixed` strategy. The average solution

		tai60a	tai80a	sko81	kra30a	ste36a	tai60b	tai80b	tai100b
`variable`	$q_{0.25}$	864	888	1016	884	991	1006	1033	1030
	$q_{0.5}$	868	890	1019	913	1010	1029	1040	1033
	$q_{0.75}$	872	893	1027	925	1021	1053	1049	1040
`k = 30`	$q_{0.25}$	647	418	576	482	574	718	724	700
	$q_{0.5}$	678	420	719	485	578	723	739	729
	$q_{0.75}$	690	558	789	490	583	728	748	761
`k =` $\mathtt{k}_{\mathtt{opt}}$	$q_{0.25}$	647	841	600	638	574	563	616	583
	$q_{0.5}$	678	844	607	646	578	566	641	599
	$q_{0.75}$	690	845	611	650	583	570	733	608

Table 7: Number of iterations for variants `var-iprob-gb`$^+$ (`variable`), `fixed(30)-iprob-gb`$^+$ (`k=30`) and `fixed(`$\mathtt{k}_{\mathtt{opt}}$`-iprob-gb`$^+$ (`k =` $\mathtt{k}_{\mathtt{opt}}$). Given are again the corresponding 0.25, 0.5, and 0.75 quantiles.

	tai60a	**tai80a**	**sko81**	**kra30a**	**ste36a**	**tai60b**	**tai80b**	**tai100b**
`ref`	**0.13**	0.13	0.09	1.60	**0.00**	1.29	**0.06**	0.14
`variable`	0.45	0.18	0.12	1.77	0.17	1.38	0.52	0.32
`k=30`	–	**0.07**	**0.06**	**1.43**	0.17	1.57	0.42	0.10
`k=k`$_{\texttt{opt}}$	0.22	**0.07**	**0.06**	**1.43**	**0.00**	**1.24**	0.25	**0.07**

Table 8: Average percentage deviation from best known solutions for $\mathcal{MM}$AS-QAP (`ref`), the best strategy `variable`, and the fixed settings. The lowest average percentage deviations on each instance are indicated in boldface.

	tai60a	**tai80a**	**sko81**	**kra30a**	**ste36a**	**tai60b**	**tai80b**	**tai100b**
r/v	0.4676	0.0366	0.9938	**0.0063**	0.0158	0.4755	$\approx \mathbf{0}$	**0.0008**
r/k$_{30}$	–	**0.0008**	0.4676	0.1545	0.6994	**0.0019**	**0.0063**	0.1545
r/k$_{\text{opt}}$	0.9938	**0.0008**	0.4676	0.1545	0.9999	0.6994	0.0783	0.0158
v/k$_{30}$	–	$\approx \mathbf{0}$	0.2810	**0.0002**	0.2810	**0.0056**	0.0783	**0.0002**
v/k$_{\text{opt}}$	0.6994	$\approx \mathbf{0}$	0.2810	**0.0002**	0.0158	0.0783	**0.0023**	$\approx \mathbf{0}$
k$_{30}$/k$_{\text{opt}}$	–	1.0000	1.0000	1.0000	0.6994	**0.0006**	0.6994	0.6994

Table 9: p-values of the two-sided Wilcoxon tests comparing the algorithms' performance. We indicate in boldface the results that are statistically significant after α-corrections according to Bonferroni.

quality reached by each of the four algorithms is given in Table 8. In Table 9 we give the p-values for the comparisons among the results, indicating those comparisons where the difference would be statistically significant. As can be seen, the performance of the iterated ants is roughly on par with the original $\mathcal{MM}$AS-QAP algorithm. Focusing on the variant where $k = 30$, we see that iterated ants perform statistically superior only on one instance, whereas they seem inferior on two others (other instances show no statistical significance). If the best setting of k would be known *a priori*, the results would be slightly more positive for the iterated ants. However, this is an ideal case and unlikely to be reachable in practice. Finally, the variant of iterated ants, where the number of solution components to be removed is kept variable, reaches worse average solution qualities than $\mathcal{MM}$AS-QAP on all instances tested, although only on three the differences are statistically significant.

Comparison based on run-time distributions. In a final step, we analyzed the run-time behavior of ia$\mathcal{MM}$AS-QAP and compared it with that of $\mathcal{MM}$AS-QAP. For doing so, we made use of the methodology based on measuring empirical run-time distributions [8]. The (qualified) run-time distribution (RTD) characterizes for a given algorithm and instance the development over time of the probability of reaching a candidate solution within a specific bound on the desired solution quality. As usual, we have chosen very high quality limits to be reached by the algorithms. In Figure 2 we show exemplary RTDs measured across 100 trials of the algorithms on several instances; 100 trials are chosen to make the RTDs reasonably stable. In the RTD plots we additionally included two exponential distributions that give an indication of whether an algorithm shows stagnation behavior or not; one exponential distribution for the original $\mathcal{MM}$AS-QAP and one for the ia$\mathcal{MM}$AS-QAP algorithm that shows best performance. (For a detailed explanation of how to check for stagnation behavior see [8, 17].) As can be seen from these plots, the iterated ants variants are somewhat prone to stagnation behaviour on instances `kra30a`, and `tai60b`, while no clear sign of stagnation is detected on the other instances. $\mathcal{MM}$AS-QAP, in contrast, apparently does not suffer from a significant stagnation behavior on any of the instances. For the algorithm variants, on which stagnation behavior is observed, the associated exponential distribution indicates the behaviour if an optimal restart-strategy or some effective diversification features would additionally be included into the algorithm. Hence, this analysis of the RTDs suggests that the ia$\mathcal{MM}$AS-QAP algorithm could for some instances strongly benefit from such strategies, contrary to $\mathcal{MM}$AS-QAP. Hence, by probably some further work, the ia$\mathcal{MM}$AS-QAP algorithm may become rather competitive to $\mathcal{MM}$AS-QAP.

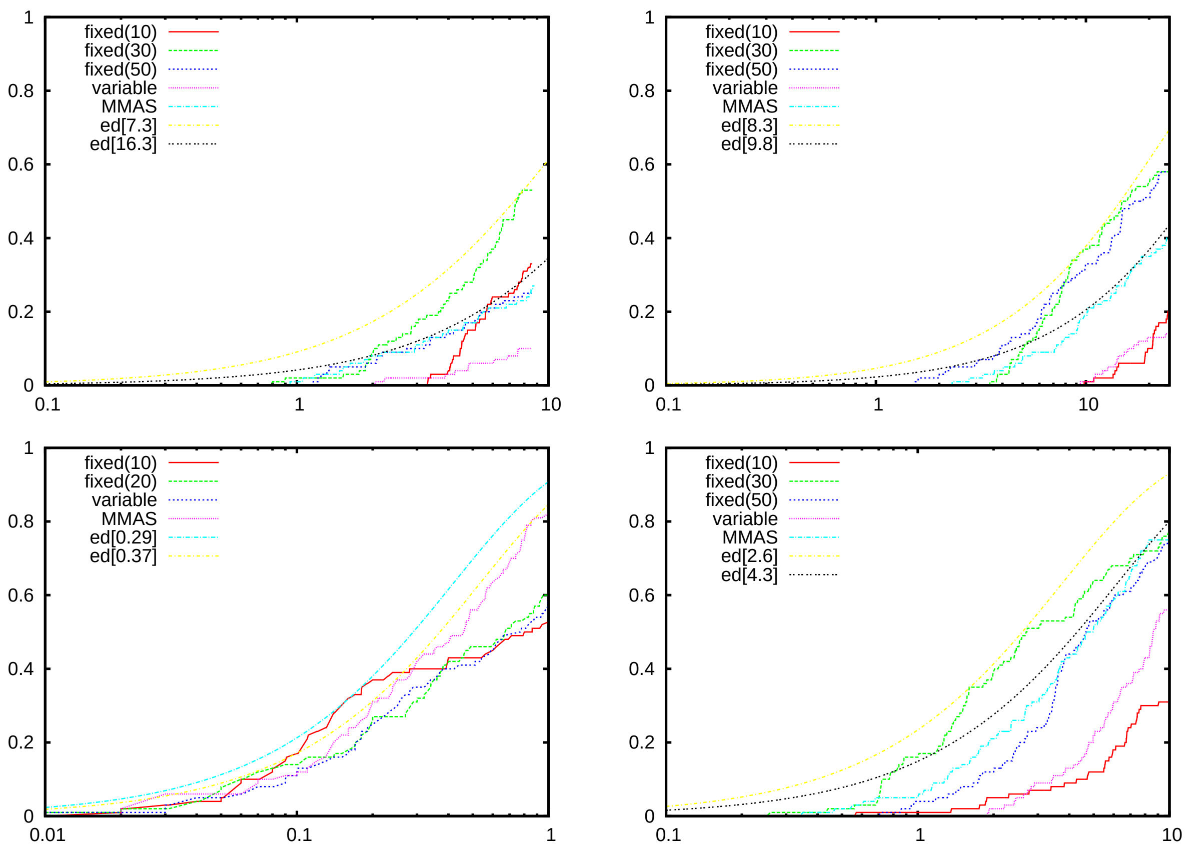

Figure 2: RTDs measured with respect to high quality solutions for various of the QAP instances tested. From top left to bottom right the RTDs are given for instances `tai60a` (1.5%), `sko81` (0.1%), `kra30a` (optimum), `tai60b` (best–known solution value); in parenthesis are given the bounds on the solution quality for which the RTDs were generated. The given exponential distributions in each plot indicate the idealised performance with an optimal restart strategy ($ed[d]$, where the value for d refers to a functional form of $ed(x) = 1 - 2^{-x/d}$).

4 Discussion and conclusions

In this paper we examined the possibility of integrating the essential ideas of the iterated greedy method into ACO algorithms. The computational results of this adaptation show that the introduction of the idea of using partial solutions to seed the solution construction of ants does not necessarily improve the performance of a state-of-the-art ACO algorithm. This can be considered a kind of negative result on the combination of two SLS methods into a higher performing algorithm, a combination that looked promising at first sight.

In fact, this negative result is a bit in contrast to the positive results reported by Acan for an earlier, similar approach that mainly differs in the way partial solutions are chosen by the ants. However, these positive results were reported for a version of $\mathcal{MM}$AS that performed rather poorly and that the reported improved performance of the extended algorithm is still far away from the results reported here. Similarly, earlier research reported positive results for an “iterated ants” algorithm for the unweighted set covering problem, where standard local search algorithms appear not to be very high performing (for the weighted set covering problem, the iterated ants algorithm performed sometimes worse than the underlying ACO algorithm) [10]. While the integration of the iterated ants concept into a state-of-the-art ACO algorithm failed to yield significant improvements, these two researches indicate that the exploitation of the iterated ants idea could be more promising if either the final solution quality reached by the ACO algorithm is still far from optimal or no effective local search for a problem exists.

Acknowledgments. Thomas Stützle acknowledges support of the Belgian FNRS, of which he is a research associate.

References

[1] A. Acan. An external memory implementation in ant colony optimization. In M. Dorigo et al., editors, *ANTS'2004, Fourth Internatinal Workshop on Ant Algorithms and Swarm Intelligence*, volume 3172 of *LNCS*, pages 73–84. Springer Verlag, 2004.

[2] A. Acan. An external partial permutations memory for ant colony optimization. In G. Raidl and J. Gottlieb, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3448 of *LNCS*, pages 1–11. springer-lncs, 2005.

[3] M. J. Brusco, L. W. Jacobs, and G. M. Thompson. A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set covering problems. *Annals of Operations Research*, 86:611–627, 1999.

[4] R. E. Burkard, E. Çela, P. M. Pardalos, and L. S. Pitsoulis. The quadratic assignment problem. In P. M. Pardalos and D.-Z. Du, editors, *Handbook of Combinatorial Optimization*, volume 2, pages 241–338. Kluwer Academic Publishers, 1998.

[5] E. Çela. *The Quadratic Assignment Problem: Theory and Algorithms.* Kluwer Academic Publishers, 1998.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, USA, 2004.

[7] P. Hahn. QAPLIB - a quadratic assignment problem library. `http://www.seas.upenn.edu/qaplib`, 2006. Version visited last on 15 February 2006.

[8] H. H. Hoos and T. Stützle. *Stochastic Local Search—Foundations and Applications.* Morgan Kaufmann Publishers, 2004.

[9] L. W. Jacobs and M. J. Brusco. A local search heuristic for large set-covering problems. *Naval Research Logistics*, 42(7):1129–1140, 1995.

[10] Lucas Lessing. Ant colony optimization for the set covering problem. Master's thesis, Intellectics Group, Computer Science Department, Darmstadt University of Technology, feb 2004.

[11] E. Marchiori and A. Steenbeek. An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In S. Cagnoni et al., editors, *Real-World Applications of Evolutionary Computing, EvoWorkshops 2000*, volume 1803 of *LNCS*, pages 367–381. Springer Verlag, 2000.

[12] Rubén Ruiz and Thomas Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, In press.

[13] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[14] T. Stützle. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System for the quadratic assignment problem. Technical Report AIDA–97–4, FG Intellektik, FB Informatik, TU Darmstadt, July 1997.

[15] T. Stützle and M. Dorigo. ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 33–50. McGraw Hill, UK, 1999.

[16] T. Stützle and H. H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[17] T. Stützle and H. H. Hoos. Analysing the run-time behaviour of iterated local search for the travelling salesman problem. In P. Hansen and C. C. Ribeiro, editors, *Essays and Surveys on Metaheuristics*, pages 589–611. Kluwer Academic Publishers, 2001.

[18] T. Stützle and H.H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and local search for combinatorial optimization problems. In S. Voss et al., editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 137–154. Kluwer Academic Publishers, 1999.