A Study of Stochastic Local Search Algorithms for the Quadratic Assignment Problem

Mohamed Saifullah Bin Hussin, Thomas Stützle and Mauro Birattari,
IRIDIA, CoDE, Université Libre de Bruxelles, Belgium

Abstract

In this article we present preliminary results on the comparison of stochastic local search algorithms for the Quadratic Assignment Problem. Our experimental comparison is based on re-implementations or adaptations of some high performing stochastic local search algorithms and it is done using default configurations and configurations tuned by F-race, a tool for the automatic tuning of parameters of stochastic algorithms.

1 Introduction

The Quadratic Assignment Problem (QAP) is an $\mathcal{NP}$-hard combinatorial optimisation problem. A QAP instance is defined by n items and n locations and two $n \times n$ matrices A and B, where a_{ij} is the distance from location i to location j, and b_{rs} is the flow between item r and item s. The objective of the QAP is to find an assignment of items to locations, such that each item is assigned to exactly one location, no location is assigned more than one item, and the sum of all products of the pairs of distances $\times$ flows is minimised.

The largest instance from QAPLIB, a widely used benchmark resource for research on the QAP, that has been solved by an exact algorithm is the $ste36a$ instance with $n = 36$; its solution took about 186 hours on a Pentium III 800 MHz CPU. The $ste36a$ instance, however, can be solved by state-of-the-art stochastic local search (SLS) algorithms in a few seconds on similar speed computers. Hence, given their clear advantage over exact algorithms, it is clear that there has been an enormous interest in SLS algorithms for the QAP. However, due to different experimental protocols, from the QAP literature the best performing algorithms for the various QAP instance classes cannot be fully reliably determined. Therefore, one main goal of our research is to investigate the relative performance of SLS algorithms for various instance classes of the QAP. In this extended abstract, we present some preliminary results of our research efforts where we compare SLS algorithms based on several of the most prominent SLS methods. For each of these methods we use for our experiments one version with default parameter settings and one with the parameters tuned by F-race [2, 1].

2 SLS Methods

Each of the SLS methods we implemented is based on a local search that uses the 2-exchange neighbourhood, where two solutions are neighboured if they differ in the assignment of exactly two items.

Simulated Annealing (SA) is an SLS method that is inspired from the physical annealing process. We adopt the SA variant proposed by Connolly [3] in addition to implementing a rather basic SA variant using a standard cooling schedule. In the tuning both, Conolly's and the 'usual' SA variants were considered together.

Tabu Search (TS) uses the search history to escape from local minima and to implement an explorative search strategy. We reimplemented a TS variant of Taillard [7], the robust Tabu Search (RTS) algorithm. We tuned three parameters: the number of iterations before the secondary aspiration criterion that implements a search diversification is applied, and the interval from which the tabu tenure is chosen periodically at random; this interval is defined by a starting tabu list size, and the interval length.

Iterated Local Search (ILS) is a simple, yet effective metaheuristic that was derived from the idea of iteratively building a sequence of local minima to efficiently sample local optima. We implemented various possible operators for an ILS for the QAP. For the perturbation fixed perturbation size schemes and schemes from Variable Neighbourhood Search were considered; the possibilities for the local search range from first-improvement, best-improvement algorithms to short tabu search runs. For the acceptance criterion, we considered five different possibilities ranging from very greedy to very permisive ones. (Note that various of the ILS variants tested in [5] are not included in this present study.)

Ant Colony Optimisation (ACO) is a metaheuristic that is inspired by the pheromone trail laying and following behaviour of real ant colonies when foraging. We adopt the $\mathcal{MAX}-\mathcal{MIN}$ Ant System, ($\mathcal{MM}$AS), an improvement over the Ant System by Stützle and Hoos [6]. We tune three parameters for $\mathcal{MM}$AS; the local search option to be used (essentially the same options as available for the ILS), the number of ants, and the pheromone evaporation factor.

Memetic Algorithms (MAs) are a population-based search technique that can be seen as a skilled combination of evolutionary algorithms with problem-specific operators, most notably local search. Here, we use a re-implementation of a MA by Merz and Freisleben [4]. We tune five parameters which are the local search option to be used (the same options as for the ACO algorithm have been considered), the population size, the mutation factor, the child factor, and the mutation strength.

3 Experimental setup and results

We have generated a new set of benchmark instances for the QAP. These instances are composed of all combinations of three different possibilities for the distance matrix—random distance matrices (R), Manhattan distances from a grid (G), and Euclidean distances (S)—and two possibilities for the flow matrix—random flows (R) and structured flows (S). This results in six instance classes: GR, GS, RR, RS, SR, and SS. We use the F-race procedure using a sampling-based approach for the definition of the algorithm configurations to be considered to tune each of the above mentioned five SLS methods on each instance class [1]. On each instance, all of which are of size $n = 80$, the SLS algorithms have available 27 seconds on an AMD Opteron 244 1.75 GHz processor with 1 MB L2-Cache and 2GB RAM, running under the Rocks Cluster GNU/Linux. (The time corresponds roughly to a run of RTS of $1000 \cdot n$ iterations. For each instance class, 200 instances were available for tuning. After the F-Race has finished, 100 fresh instances from each instance class are run independently using default and tuned parameter settings for comparing the performance of the SLS algorithms.

We measure the performance of the different SLS algorithms running once each algorithm on each test instance. We then compute the average value of the returned solutions on each instance class when using different SLS algorithms. Here, we only present results at a high-level comparing the performance of the algorithms on the various instance classes. In Tables 1 and Table 2, we give the computational results for the default parameter settings and the winning parameter settings, respectively. In particular, for each algorithm and instance class (this is done independently for default and tuned parameter settings), we compute the average solution value. In the tables is given the percentage excess of each algorithm over the algorithm that obtained the lowest average solution quality. (Note that when comparing for each SLS algorithm and instance class pair the default and the tuned parameter settings, except for two such pairs, always a statistically significant difference in average solution quality was observed; that is, the tuned versions are in almost all cases statistically better than the defaults. In part, these differences were quite substantial.)

Considering default configurations, $\mathcal{MM}$AS and RTS showed the best

Table 1: Distance from the best solution by using default parameter settings

Instance Class	$\mathcal{MM}$AS	ILS	MA	RTS	SA
GR	**0.00**	*0.44*	0.09	0.03	0.14
GS	0.31	*2.90*	0.79	**0.00**	1.03
RR	0.41	*0.94*	0.62	**0.00**	0.84
RS	2.84	*6.11*	4.45	**0.00**	5.56
SR	**0.00**	0.47	0.09	*0.65*	0.11
SS	**0.00**	*1.75*	0.14	1.62	0.26

Table 2: Distance from the best solution by using winning parameter settings

Instance Class	$\mathcal{MM}$AS	ILS	MA	RTS	SA
GR	0.01	0.07	**0.00**	0.01	*0.16*
GS	0.10	0.33	0.17	**0.00**	*1.16*
RR	0.10	0.14	0.05	**0.00**	*0.65*
RS	0.27	0.79	0.14	**0.00**	*4.73*
SR	0.04	0.16	**0.00**	0.17	*0.18*
SS	0.16	*0.49*	**0.00**	0.38	0.29

performance in most of the cases. $\mathcal{MM}$AS achieves the best results for instance classes GR, SR, and SS, while RTS achieves the best results for instance classes GS, RR, and RS. In general, however, the dependence of the performance of RTS on the instance class seems to be higher than for $\mathcal{MM}$AS. While $\mathcal{MM}$AS is either the best or second best algorithm, RTS, performs very poorly for the instance classes SR and SS. ILS seems to be the lowest performing algorithm among the chosen ones. On most instance classes, except in case of instance class SR, it is the last ranked algorithm.

Considering the tuned versions, MA achieves the best performance for instance class GR, SR, and SS, while RTS achieves the best results for instance classes GS, RR, and RS. In almost all cases, SA achieves the worst performance compared to other SLS algorithms. The exception is for instance class SS where ILS is worst. Overall, MA achieves the best results compared to other SLS algorithms.

4 Conclusions

The best performing algorithms for the various instance classes were found to be the tuned version of either RTS or MA. Among the default settings, among the best ranked ones was the $\mathcal{MM}$AS, which indicates that it offers rather robust performance across a wide set of different types of instances. When considering the ranking of the different algorithm variants, apparently the MA was profiting most from the tuning in the sense that for most instance classes it could improve its relative ranking when compared to the other algorithms. Finally, the performance of RTS was rather dependent on the instance class. While for several ones it was among the top two ranked algorithms, on two instance classes it was among the two worst ones.

References

[1] P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo. Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In *4th International Workshop on Hybrid Metaheuristics, Proceedings, HM 2007*, LNCS. Springer Verlag, Berlin, 2007.

[2] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 11–18, 2002.

[3] D. T. Connolly. An improved annealing scheme for the QAP. *European Journal of Operations Research*, 46(1):93–100, 1990.

[4] P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms and greedy operators for graph bi-partitioning. *Evolutionary Computation*, 8(1):61–91, 2000.

[5] T. Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operations Research*, 174(1):1519–1539, 2006.

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

[7] É. D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17(4–5):443–455, 1991.