2011 30th International Conference of the Chilean Computer Science Society

Evaluating memory schemas in a Memetic Algorithm for the Quadratic Assignment Problem

Hugo Meneses
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, USACH
Santiago, Chile
Email: hugo.menesesp@usach.cl

Mario Inostroza-Ponta*
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, USACH
Santiago, Chile
Email: mario.inostroza@usach.cl
corresponding author

***Abstract*—The use of memory schemas in metaheuristics has been an important technique to improve the performance of the algorithms in order to perform a better search in the solution space of a given problem. There are some techniques that are inherent to the structure used, like population in population-based metaheuristics, or the tabu list in Tabu Search. It is common that these techniques are used alone or combined, towards the construction of a better algorithm. In this work, we present the inclusion of several memory schemas in a Memetic Algorithm for the Quadratic Assignment Problem. The original MA algorithm already has good performance when compared with two other state of the art population-based metaheuristic algorithms. As a result of the memory schemas used, we were able to improve the quality of the solution generated by the Memetic Algorithm. The results are compared using the quality of the solutions and the success rate using the Copeland index.**

***Keywords*-Memetic Algorithm; Quadratic Assignment Problem; Memory Schemas**

I. Introduction

The Quadratic Assignment Problem (QAP) is considered one of the hardest combinatorial optimization problems. It can only be solved for instances of about 30 objects. There are several works that attempt to create and improve algorithms to find better solutions. In the QAP it is necessary to assign n facilities into n locations. A solution can be represented by a permutation π, where each element $\pi(i) = k$ indicates that a facility i has been assigned to location k. The problem is represented using a matrix D for the distances between locations, matrix F for the flows between facilities and S_n representing all permutations of size n. A solution for the QAP consists of finding a permutation π such that the fitness function

$$C(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} F(i,j) \times D(\pi(i), \pi(j)); \pi \in S_n \qquad (1)$$

is minimized. QAP was first presented as an economical problem [1] and has been proved to belong the *NP-Hard* class of problem [2]. Because of that several attempts to find good algorithms have been performed. In [3], the authors show several techniques used in different works to deal with QAP. Among them, population-based metaheuristics like Memetic Algorithms [4]. A Memetic Algorithm (MA) uses concepts from evolutionary computing and local search strategies. The *memetic* denomination comes from the word *meme*, that is analogue to *gene* in cultural evolution. In a MA there is a population of agents that contain one or more candidate solutions of the problem. This population can have a specific structure or only a set of agents without relationship between them. The size of the population is a parameter to be set during the parametrization step. However, it is known that small populations have the problem of premature convergence to local optimal solution. On the other hand, large populations can become unmanageable, specially when the algorithm operators or the fitness function are too computationally expensive. A technique used to avoid premature convergence in small populations is the use of additional memory schemas [5], [6].

In this work we compare the performance of the combination of 4 memory schemas, applied to the memetic algorithm for QAP presented in [6]. The results are presented and compared in terms of quality of solution using instances from the literature. Additionally, we performed a pairwise comparison between the algorithms obtained.

II. Memory schemas

The use of memory is common strategy to guide an algorithm during the execution, in order to avoid the repetition of movement or solutions, and to navigate the solution space in a better way. In particular, population based metaheuristics make use of different memory schemas, sometimes in an explicit manner (like tabu list in Tabu Search) or implicit (population in Genetic Algorithms). There are several works that aim to exploit the use of memory as a strategy in the design of an algorithm. In [7] a memory is used to store solutions and to be re-inserted into population when environment changes. Same situation is confronted by [8] using a memory of variable size. In [9] a memory is used to store solutions in order to detect the repeated ones to be replaced.The authors in [10] uses a population of ants and

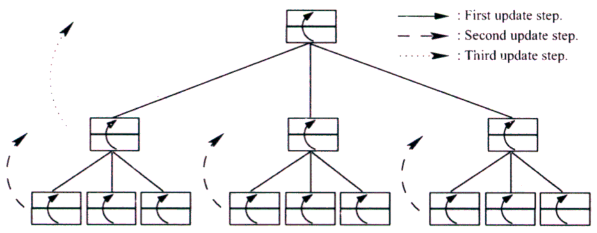

Figure 1. Population structured used in the MA. Arrows show the update procedure performed in three steps: in each agent, in subpopulation one, two and three, and finally in subpopulation zero.

store pheromone traces to deal with QAP. Same problem is treated in [11], saving common assignations between parents to reduce search space.

The memetic algorithm presented by Inostroza-Ponta [6] uses a structured hierarchical population of 13 agents, each of them containing five working solutions and one pocket solution. The pocket solution corresponds to the best solution found by that agent in a certain generation. Figure 1 shows the population organization.

The population is organized in four overlapped sub populations with each of them made of 3 supporters agents and one leader agent. The leaders of sub populations 1, 2 and 3 are the supporters of sub population 0. The hierarchy of the population is kept by two update functions shown on the algorithm in Figure 2 (**updateAgent()** and **updatePop()**). This structure was first used in [5] for the solution of the Travelling Salesman Problem, with only one solution per agent. The **updateAgent()** function replaces the pocket solution if one of the working solutions has a better cost. The **updatePop()** function works in two step: first it replaces the pocket solution of the leader agent in each subpopulation if there is one of the pocket solutions of the supporters has a better cost, and second, it replaces the pocket solution of the leader agent of the population if there is a better solution among the leaders. These updates steps are indicated by arrows in Figure 1.

The algorithm implemented is shown on Figure 2. After the population is initialized (**InitializePop()**), for each agent the algorithm applies the population operators **selectParents()** and **crossover()** to generate an offspring. On that offspring the algorithms applies a Tabu Search implementation of pairwise interchange heuristic (**swapTS()**). The details of these operators can be found in [6].

The algorithm presented in Figure 2 uses three memory schemas. First, the structure of the population corresponds to a long term memory schema, since it contains a hierarchy among the members of the population which is maintained during the whole execution of the algorithm. It results on keeping track of the best solution found so far by the algorithm. We call this memory schema **SP**. Second, the Tabu List implemented in the local search, that keeps track of the forbidden movements in the pairwise interchange heuristic.

```
MA(F,L)
$n$: dim(F) //number of objects
$m$: dim(L) //number of locations
  $pop$ = initializePop()
  REPEAT
     FOR $i$=1 TO 12
        selectParents($i$,$parent_1$,$parent_2$)
        $offspring$ = crossover($parent_1$,$parent_2$)
        swapTS($offspring$)
        updateAgent($offspring$,$i$)
     END FOR
     updatePop($pop$)
     IF popConverged($pop$)
        $pop$ = restartPop($pop$)
     END IF
  UNTIL $max\_number\_of\_generations$
END ALGORITHM
```

Figure 2. Pseudo-code of the Memetic Algorithm implemented for the QAP.

Finally, the Tabu List is kept in the same subpopulation in order to keep track of the history of crossovers. This Tabu List is restarted when there is a restart of the population.

In this work we evaluate the inclusion of other memory schemas and their combination:

- **FTL**: a Fixed Tabu List using a variation of the original Tabu Search presented by Glover [12] where the size of the tabu list is fixed. We used a size of 2 X $maxIter$.
- **VTL**: Variable Tabu list. Similar to the above, but in this case it uses a tabu list of variable size within a range. Already implemented in the algorithm. It uses a range of $[0.1, 0.5]$ X $maxIter$ for subpopulation 0 and $[0.1, 0.3]$ X $maxIter$ for subpopulations one, two and three.
- **RS**: Reduced Search. Previously used in [11]. Each individual saves the equal positions among its parents, and are initially prohibited to be used in the local search. It reduces the solution search space.
- **SD**: Steepest Descent. It performs a greedy 2-opt local search within the neighbourhood of the solution. Each time a new best solution is found, the **SD** is applied, until it cannot be further improved.

The combination of the above memory schemas produces eight different memetic algorithms. It is clear that **FTL** and **VTL** cannot be combined. Table I shows the different version of the algorithm produced. The version using only **VTL** correspond to the original memetic algorithm used in [6]. The structured population and the shared Tabu List are kept as in the original work.

III. Results and Discussion

In order to test the performance of each of the algorithms with different combination of memory schemas, we used 30 instances (Table II of the QAP taken from the literature

Table I
ALGORITHMS CREATED WITH MEMORY SCHEMAS. THE **SP** INDICATES THAT MEMORY FROM THE STRUCTURE POPULATION IS ALWAYS CONSIDERED. IN BOLD WE INDICATE THE ORIGINAL ALGORITHM

	FTL	VTL
-	SP-FTL	**SP-VTL**
RS	SP-FTL-RS	SP-VTL-RS
SD	SP-FTL-SD	SP-VTL-SD
RS-SD	SP-FTL-RS-SD	SP-VTL-RS-SD

and that were also used in the original algorithm [6]. Each algorithm is run 10 times on each instance. We use a computer with AMD Opteron 6128, 2 Ghz and 16 GB of RAM. The algorithms do not take advantage of the parallel capabilities of the computer. For the local search, we used a *maxIter* of 100, which indicates the number of iterations of the local search without improvements.

Table II
QAP INSTANCES USED FOR EVALUATION PURPOSES. SECOND COLUMN SHOWS THE SIZE OF THE INSTANCE n AND THIRD COLUMN SHOWS THE BEST KNOWN VALUE (BKV) TAKEN FROM [13]

Instance	**n**	**BKV**
chr25a	25	3796
kra30a	30	88900
nug30	30	6124
ste36a	36	9526
sko49	49	23386
sko56	56	34458
sko64	64	48498
sko72	72	66256
sko81	81	90998
sko90	90	115534
sko100a	100	152002
tai20a	20	703482
tai25a	25	1167256
tai30a	30	1818146
tai35a	35	2422002
tai35b	35	283315445
tai40a	40	3139370
tai40b	40	637250948
tai50a	50	4938796
tai50b	50	458821517
tai60a	60	7205962
tai60b	60	608215054
tai80a	80	13515450
tai80b	80	818415043
tai100a	100	21052466
tai100b	100	1185996137
tho150b	150	498896643
wil50	50	48816
tai256c	256	44759294
tai150b	150	498896643

The performance of the algorithms is compared in terms of quality of the solutions (%GAP from the best known value reported in the literature [13], eq 2).

$$\%GAP = \frac{Value_{obtained} - BKV}{BKV} \times 100\% \tag{2}$$

Additionally, we use the Copeland index [14] to compare the success rate of the algorithms in a pairwise manner using equation 3. In equation 3, σ_m is the Copeland index of algorithm m, m' represents others algorithms, $c_{mm'}$ the number of times that m defeats m' and $c_{m'm}$ the opposite.

$$\sigma_m = \sum_{m'} (c_{mm'} - c_{m'm}) \tag{3}$$

The results are shown in Table III. Each column represents one of the algorithms. The last two rows show the average %GAP and the Copeland index respectively. All algorithms show a better performance than the original algorithm used in [6]. In particular the best performance is reached by the combination of three memory schemas: Variable Tabu List, Reduced Search and Steepest Descent. This algorithm not only reach the best %GAP but also it gets the best Copeland index, which shows that its success rate is more significant than the rest of the algorithms. The Reduced Search restricts the initial search space giving a better start point and Steepest Descent makes a final refinement to the improved solution, giving the chance of get better solutions than just using Tabu Search.

Table IV shows the average error and Copeland indexes of the algorithms organized by memory schema. It can be seen that in average the best performance are obtained when the algorithms use RS (0.138, 28) and SD (0.139, 9). Even more the better performance overall is obtained when RS and SD are combined (0.136, 20). This confirms the original goal of this work that the combination of memory schemas helps the algorithm to reach better solutions.

Table IV
AVERAGE ERRORS AND COPELAND INDEXES

	Error	Copeland
VTL	**0,138**	**13**
FTL	0,140	-13
RS	**0,138**	**28**
no RS	0,140	-28
SD	**0,139**	**9**
no SD	0,140	-9
RS-SD	**0,136**	**20**
no RS-SD	0,140	-20

Further tests are performed in order to evaluate the performance of the algorithms compared with other state of the art population-based algorithms. We compared against an ant-colony algorithm presented in [15] and another memetic algorithm presented in [11]. Tables V and VI show the results on the instances used by the original works respectively. It is clear to see that our algorithm outperforms both of the previous work. The work of Merz [11] uses memory, saving the common assignments between parents in crossover to reduce the solution space in local search. In the case of Demirel [15], they use the pheromone trail of the ants as memory structure. Additionally, they use Simulated Annealing as a local search strategy.

IV. CONCLUSION

In this paper we have presented a modification of a previously presented Memetic Algorithm for the Quadratic Assignment Problem. The modification is based on the exploration of the use of extra memory schemas in the algorithm. These memory schemas are not new, but we combined them to test if their sinergy produces a better algorithm. The combination of the memory schemas produce seven different versions of the algorithm. Each of them was tested using instances taken from the literature. After the computational tests, we selected a best combination of memory, based on both the quality of the solution (measure by the average GAP from the Best Known Value) and the success rate (measured

Table III
RESULTS TABLE. BOLD VALUES INDICATES ALGORITHMS THAT FIND THE BEST SOLUTION FOR THAT INSTANCE. THE SECOND LAST ROW SHOWS THE AVERAGE %GAP AND THE LAST ROW SHOWS THE COPELAND INDEX.

	SP							
Instance	VTL	VTL-RS	VTL-SD	VTL-RS-SD	FTL	FTL-RS	FTL-SD	FTL-RS-SD
chr25a	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
kra30a	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
nug30	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
ste36a	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	0,010	**0,000**
sko49	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
sko56	0,002	**0,000**	0,001	0,002	**0,000**	**0,000**	**0,000**	**0,000**
sko64	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
sko72	0,006	**0,000**	**0,000**	0,007	0,014	0,006	**0,000**	0,006
sko81	**0,007**	0,016	0,010	0,013	0,016	0,024	0,020	0,019
sko90	0,025	**0,003**	0,018	0,004	0,018	0,010	0,015	0,008
sko100a	0,041	0,040	0,024	0,030	0,037	0,029	**0,015**	0,030
tai20a	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai25a	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai30a	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai35a	0,051	**0,017**	0,040	0,019	0,034	0,041	0,053	0,045
tai35b	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai40a	0,290	**0,236**	0,307	0,262	0,244	0,308	0,331	0,352
tai40b	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai50a	**0,495**	0,660	0,565	0,620	0,683	0,636	0,625	0,634
tai50b	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai60a	0,661	0,698	0,675	0,660	0,694	0,695	**0,659**	0,688
tai60b	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai80a	0,782	0,901	0,840	**0,733**	0,820	0,817	0,778	0,846
tai80b	0,133	0,059	0,124	0,015	0,113	0,106	0,010	**0,005**
tai100a	0,727	0,770	0,717	0,710	0,703	**0,674**	0,741	0,677
tai100b	0,117	0,092	0,133	0,071	0,120	0,108	0,177	**0,060**
tho150b	0,101	0,147	**0,082**	0,130	0,110	0,083	0,140	0,094
wil50	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**
tai256c	**0,093**	0,105	0,111	0,116	0,102	0,100	0,111	0,105
tai150b	0,534	**0,470**	0,667	0,592	0,555	0,569	0,497	0,608
Average	0,136	0,140	0,144	**0,133**	0,142	0,140	0,139	0,139
Copeland	-4	3	-3	**17**	-13	5	-8	3

Table V
COMPARISON AGAINST MERZ ET AL [11]

Instance	**Merz2000**	**SP-VTL-RS-SD**
nug30	0,004	**0,000**
ste36a	0,045	**0,000**
sko100a	0,096	**0,030**
tai60a	1,351	**0,660**
tai60b	**0,000**	**0,000**
tai80a	1,423	**0,733**
tai80b	**0,004**	0,015
tai100a	1,110	**0,710**
tai100b	**0,038**	0,071
tho150b	0,361	**0,130**
tai256c	**0,070**	0,116
tai150b	**0,361**	0,592
Average	0,405	**0,255**

Table VI
COMPARISON WITH RESULTS OBTAINED BY DEMIREL ET AL. [15]

Instance	**ACSA**	**SP-VTL-RS-SD**
chr25a	1,245	**0,000**
kra30a	0,776	**0,000**
sko49	0,027	**0,000**
sko56	**0,001**	0,002
sko64	0,041	**0,000**
sko72	0,221	**0,007**
sko81	0,077	**0,013**
sko90	0,136	**0,004**
tai20a	0,255	**0,000**
tai25a	0,098	**0,000**
tai30a	0,454	**0,000**
tai35a	0,440	**0,019**
tai35b	0,038	**0,000**
tai40a	0,476	**0,262**
tai40b	0,487	**0,000**
tai50a	**0,573**	0,620
tai50b	0,248	**0,000**
tai60a	1,001	**0,660**
tai60b	0,226	**0,000**
tai80a	**0,677**	0,733
tai80b	0,821	**0,015**
wil50	0,088	**0,000**
Average	0,382	**0,106**

using the Copeland index). This algorithm was then tested against two other metaheuristics algorithms and in both cases our algorithm is able to outperformed them.

These results allow us to believe that the use of memory can be further explored in other metaheuristics and other optimization problems.

REFERENCES

[1] T. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, pp. 53–76, 1957.

[2] S. Sahni and T. Gonzalez, "P-complete approximation problems," *J. ACM*, vol. 23, pp. 555–565, July 1976.

[3] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, pp. 657–690, 2007.

[4] P. Moscato and C. Cotta, "An introduction to memetic algorithms," *Revista Iberoamericana de Inteligencia Artificial*, vol. 19, pp. 131–148, 2003.

[5] L. Buriol, P. Franca, and P. Moscato, "A new memetic algorithm for the asymmetric traveling salesman problem," *Journal of Heuristics*, vol. 10, pp. 483–506, 2004.

[6] M. Inostroza-Ponta, "An integrated and scalable approach based on combinatorial optimization techniques for the analysis of microarray data," Ph.D. dissertation, The University of Newcastle, Callaghan, NSW 2308, March 2008.

[7] S. Yang and X. Yao, "Population-based incremental learning with memory scheme for changing environments," in *in Proc. 2005 Genetic Evol. Comput. Conf.*, 2005, pp. 711–718.

[8] A. Simoes, E. Costa, R. Pedro, and N. Q. Nora, "Variable-size memory evolutionary algorithm to deal with dynamic environments," in *In M. Giacobini et al. (Eds.): EvoWorkshops 2007, Applications of Evolutionary Computing, LNCS 4448*. Springer Verlag, 2007, pp. 617–626.

[9] M. Wiering, "Memory-based memetic algorithms," in *Proceedings of the Thirteenth Belgian-Dutch Conference on Machine Learning*. Benelearn, 2004, pp. 191–198.

[10] L. Gambardella, D. Taillard, and M. Dorigo, "Ant colonies for the qap," 1998.

[11] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, November 2000.

[12] F. Glover, "Tabu search–part i," *Informs Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[13] R. E. Burkard, S. E. Karisch, and F. Rendl, "QAPLIB A Quadratic Assignment Problem Library," *Journal of Global Optimization*, vol. 10, pp. 391–403, June 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=596060.596140

[14] E.-G. Talbi, *Metaheuristics : from design to implementation*, ser. The Sciences Po series in international relations and political economy. John Wiley & Sons, 2009, vol. 10, no. PART A.

[15] N. Demirel and M. Toksari, "Optimization of the quadratic assignment problem using an ant colony algorithm," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 427—435, 2006.