Parallel Hybrid Meta-Heuristics: Application to the Quadratic Assignment Problem

V. Bachelet* P. Preux† E-G. Talbi‡

February 27, 1996

Abstract: Meta-heuristics are search techniques that can be applied to a broad range of combinatorial optimization problems. Each meta-heuristic explores and exploits the search space in its own way. No heuristic can be better than any heuristic on a wide spectrum of problems. To make the search more efficient and robust, hybridization of heuristics should be used.

In this paper, we present an ongoing research on parallel hybrid heuristics. The Quadratic Assignement Problem is used as a testbed problem. We present the performance of different meta-heuristics and their hybridization on standard problems taken from the QAPLIB library.

Keywords: Parallel heuristic, Hybrid algorithm, Quadratic assignment problem, Hill-climbing, Genetic algorithm, Tabu search.

1 Introduction

Unless P=NP, many interesting combinatorial optimization problems cannot be solved exactly within a reasonable amount of time. Consequently, heuristics must be used to solve large real-world problems.

Heuristic algorithms may be divided into two main classes. First, the *general* purpose algorithms (meta-heuristics) independant of the optimization problem and, on the other hand, the *tailored* algorithms especially designed for a given problem. As we want to avoid the intrinsic disadvantage of the algorithms of this second class (their limited applicability), our research study is only concerned with the first class of algorithms.

In the design of a heuristic search strategy, two contradictory criteria should be considered: the exploration of the search space and the exploitation of the solutions that have already been found. While hill-climbing carry to extremes the exploitation criterion, random search carry to extremes the exploration criterion. Many efficient heuristics balance the two criteria. Those widely used in the literature are: simulated annealing, tabu search and genetic algorithms.

Each meta-heuristic explores and exploits the search space in its own way. No heuristic can be better than any heuristic on a wide spectrum of problems. Instead of trying to improve diversifying heuristics and intensifying ones, a common idea is to hybridize both approaches. Hybridization aims at making both heuristics compensate themselves and federate their behaviors. To evaluate the performances of different heuristics, we use a testbed problem for our experiments: the quadratic assignment problem (QAP).

The remainder of the paper is organized as follows. In section 2, we formulate the QAP and present the main complexity results. The section 3 describes briefly the different meta-heuristics applied to the QAP. For a more detailed presentation, the reader is referred to [20]. The hybridization of the meta-heuristics will be detailed in section 4, being one of the key-points of our work. Finally, in section 5 we will present results of experiments, using different instances of the QAP. We compare the efficiency of hybrid algorithms with the best algorithms proposed so far.

*Ecole des Mines de Douai, 941 rue Charles Bourseul, BP 838, 59508 Douai Cedex, France, bachelet@lifl.fr

†Laboratoire d'Informatique du Littoral, BP 719, 62228 Calais Cedex, France, preux@lil.univ-littoral.fr

‡Laboratoire d'Informatique Fondamentale de Lille, URA CNRS 369, Cité scientifique, 59655 Villeneuve d'Ascq Cedex, France, talbi@lifl.fr

2 The quadratic assignment problem

The QAP represents an important class of combinatorial optimization problem with many applications in different domains (facility location, data analysis, task scheduling, ...). The first formulation was given by Koopmans and Beckmann in 1957 [13].

The QAP can be defined as follows:
Given:

- a set of objects $O = \{O_1, O_2, ..., O_n\}$,
- a set of locations $L = \{L_1, L_2, ..., L_n\}$,
- a flow matrix c, where each element c_{ij} denotes a flow cost between the objects O_i and O_j,
- a distance matrix d, where each element d_{kl} denotes a distance between location L_k and L_l,

find an object-location bijective mapping $M : O \longrightarrow L$, which minimizes the objective function f:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}.d_{M(i)M(j)}$$

The QAP is NP-hard [7]. Finding an ϵ-approximate solution is also NP-complete [22]. This fact has restricted exact algorithms (such branch and bound) to small instances ($n < 20$). Therefore, many heuristics have been proposed in the literature. An extensive survey and recent developments can be found in [18].

When the size of an instance becomes larger ($n > 50$), and for randomly generated matrices c and d, the difference between the worst solution and the best solution is very small. The probability that this difference is 0 tends towards 1 when the size of the instance tends towards infinity [21]. This asymptotic property renders the design of the test procedure a tricky task.

To evaluate the difficulty of an instance, its size is an important parameter, but it is not sufficient. Vollmann and Buffa introduce the notion of *flow dominance* [27]. It is the coefficient of variation fd of the matrix c.

$$fd(c) = 100 \times \frac{s}{m}$$

where m is the average of the c_{ij} 's:

$$m = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}}{n^2}$$

and s their standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij} - m)^2}{n^2}}$$

High values indicate that few elements dominate the matrix c. This means that we have very small local optima and a few large local optima, which turns out to be a difficult search space for a local search algorithm. This value has a large impact on the results obtained by the CRAFT local search algorithm [2].

Many encoding schemes may be used to represent a solution of the QAP. In all the algorithms, we use the same representation which is based on a permutation of n integers:

$$p = (l_1, l_2, ..., l_n)$$

where l_i denotes the location of the object O_i.

In all the heuristics, we use the same basic movement: a pair exchange in which two objects of a permutation are swapped. Hence, we have the same neighborhood for all local search based algorithms.

Let's see how the evaluation of the neighborhood can be done in $O(n^2)$ operations, if we assume that the matrix c and d are symetrical and null-diagonal, which is the case in general.

If p is the current solution, $p_{rs} = q$ represents the solution obtained after an exchange of the objects r and s. $\Delta_p(rs)$ represents the variation of the objective function ($\Delta_p(rs) = f(p_{rs}) - f(p)$). We have [25]:

$$\Delta_p(rs) = 2 \sum_{i=1, i \neq r,s}^{n} (d_{is} - d_{ir})(c_{p(i)p(s)} - c_{p(i)p(r)})$$

$\Delta_p(rs)$ can be evaluated in $O(n)$ operations. If we swap the objects u and v, we will obtain:

$$\Delta_q(uv) = \Delta_p(rs) + 2(d_{ru} - d_{rv} + d_{sv} - d_{su})(c_{q(r)q(v)} - c_{q(r)q(u)} - c_{q(s)q(v)} + c_{q(s)q(u)})$$

which can be evaluated in a constant time. As we have n^2 neighbors, the whole neighborhood can be evaluated in $O(n^2)$ operations except for the first iteration.

3 Meta-heuristics and the QAP

Three meta-heuristics have been applied to the QAP:

- hill-climbing
- parallel tabu search
- parallel genetic algorithms

3.1 Hill-climbing

We consider a hill-climbing algorithm as a baseline search algorithm because of its simplicity (cheap in its space and time requirements, and easy to design).

The hill-climbing (HC) algorithm starts with a feasible solution, and tries to improve it by local moves. A move is selected, the cost change of the move is evaluated, and if the change is positive the move is accepted and a new solution is generated. This process is repeated until there is no moves that enhance the objective function further. When this occurs, a local minimum has usually been found, rather than a global minimum.

Many versions of hill-climbing algorithms may be considered. They differ in the transformation/replacement strategy that is used:

- **stochastic**: the moves are operated randomly. Only a portion of the neighborhood is generated.
- **first better**: the neighbors are generated sequentially. The first neighbor which improve the current solution is selected. In the worst case, all neighbors are generated.
- **best**: all the neighbors are generated. The best one is selected.

For the QAP, as the evaluation of the whole neighborhood takes only $O(n^2)$ operations, we use the third strategy in this paper.

3.2 Parallel tabu search

The tabu search has been proposed by Glover [8]. Unlike hill-climbing as it might make a down-hill move at each iteration of the search, it selects the best neighborhood solution even if this results in a worst solution than the current one. However, this strategy may result in cycling. So a *tabu list* and an *aspiration function* are introduced to keep information about past moves. Other advanced techniques may be used such *intensification* to encourage the exploitation of a region in the search space, and *diversification* to encourage the exploration of new regions.

An extensive review of parallel tabu search may be found in [5]. They may be classified in two main categories:

- Parallel evaluation of the neighborhood.
- Multiple tabu search processes, independant or cooperative.

In this paper, a straightforward approach has been used to introduce parallelism in the tabu search. It consists in multiple independant tabu search algorithms running in parallel (MTS). This requires no communication between the sequential algorithms. The algorithms are initialized with different solutions. Different parameter settings are also used. The aspiration function allows a tabu move if it generates a solution better than the best found solution. The tabu list contains pairs (i, j) of objects that cannot be exchanged. The size of the list is set to the size of the instance.

3.3 Parallel genetic algorithms

Genetic algorithms (GAs) are meta-heuristics based on the natural evolution of living beings [12]. Unlike hill-climbing and tabu search which work on a single solution, GAs operate on a population of individuals (solutions). They reproduce the cycle of life: *selection*, *reproduction* (involving genetic operators to generate offsprings), and a *replacement* mechanism which stochastically retain individuals amongst the parents and the offsprings favoring the best ones. Each iteration is called a generation. The algorithm iterates until a criterion is fullfilled. The application of GAs to solve the QAP has been presented in [1][15].

The growing availability of parallel computers opens new horizons to GAs. Several parallel models of GAs have been proposed and applied to solve various problems [26]. The parallel model considered in our study is a cellular genetic approach (CGA). It is a totally distributed fine grained model. The population is mapped onto a non-complete connected graph (e.g. a torus) (fig. 1), one individual per node where an edge between two individuals represent the possibility for them to mate and reproduce. The selection is done locally in the neighborhood of each individual. The CGA is very flexible. We have implemented the CGA on various types of parallel architectures (SIMD/MIMD, tighly/loosely coupled, homogeneous/heterogeneous): 16k Maspar MP-1, 16 DEC Alpha processor farm, 128 transputer network, heterogeneous workstation clusters.

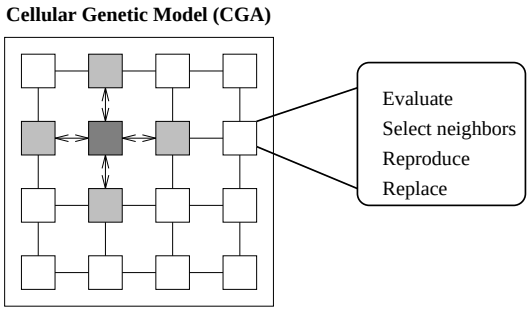

Figure 1: The cellular parallel genetic model.

As an individual has been encoded as a permutation of n integers, the mutation operator, consists in swapping two integers (pairwise exchange), which results in a new valid assignment. The crossover operator is the PMX operator [9].

Figure 2 shows the evolution of the solution quality. For the CGA, we notice that the greatest reduction in the fitness function occurs at the beginning. Thus a satisfactory solution can be obtained very quickly. After a certain number of generations, the population is quite uniform and the fitness of the population is no longer decreasing. The odds to produce fitter individuals are thus very low.

We think that no heuristic (or class of heuristics) can be better than any heuristic on a wide spectrum of problems. Recent theoretical results comfort this feeling [29]. Each heuristic explores and exploits the search space in its own way. It will perform well for certain problems, and bad for (much more) other problems. Hence we think that we should hybridize several search algorithms to make the search more efficient.

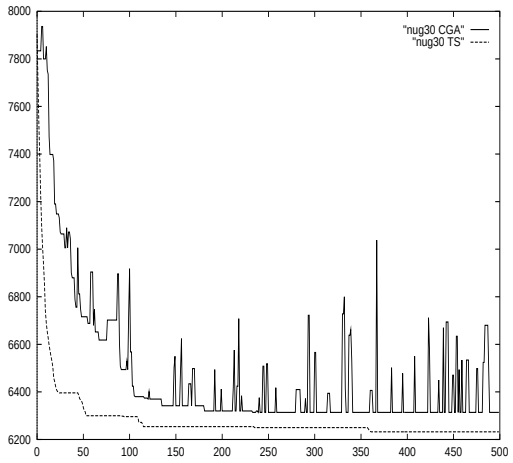

Figure 2: Convergence of the CGA (plain line) and the tabu search (dashed line) on nug30. For the CGA, the graph shows at each generation the fitness of the individual of a typical cell during one run. For the tabu search, the graph shows at each generation the fitness of the best already found solution during one run.

4 Hybrid parallel heuristics

Different hybrid methods have been identified [19]:

- Sequential hybridization
- Parallel synchronous hybridization
- Parallel asynchronous hybridization

4.1 Sequential hybridization

In a sequential hybridization, a set of algorithms is applied one after another, each using the output of the previous as its input, acting in a pipeline fashion (fig. 3).

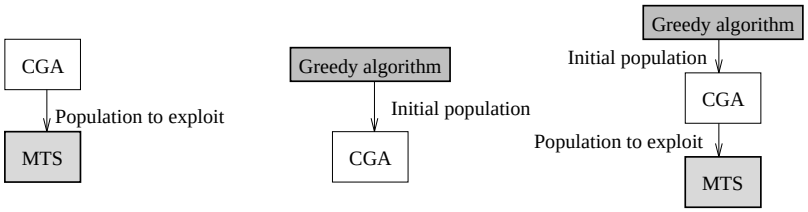

Figure 3: Sequential Hybridization. Three instances of this hybridization scheme are represented. There may be more than three algorithms to be pipelined.

The CGA is able to realize a good exploration, while the MTS is powerful in the exploitation task. Accordingly, it is very tempting to hybridize the CGA with the MTS. In this regard, we investigate sequential hybridization to get an efficient cooperation. The sequential hybrid algorithm considered here (SH) pipelines the parallel genetic algorithm (CGA) and the parallel tabu search (MTS). The MTS was used after the CGA to exploit the result of the previous exploration of the search space by the CGA. The problem of SH lies on deciding when to stop the CGA and trigger the MTS, and which individuals must be selected for the MTS to act on. The solution adopted is to wait for the stabilization of the fitness in the population of the CGA. Then the best individuals are the starting points of the MTS.

4.2 Parallel synchronous hybridization

The idea of this class of hybridization is to use some search method as an operator of another search method. Instead of using a simple operator, we use an operator which is a search algorithm that considers the individual as the origin of its search, applies the search algorithm, and finally replaces the original solution by the enhanced one. If we take a parallel genetic algorithm as the main algorithm, the blind mutation operator may be replaced by a hill-climber, a tabu search or a simulated annealing algorithm [1]. Such hybridization may lead to premature convergence.

The parallel synchronous hybrid (PSH) considered in this paper, is the CGA where a sequential tabu search algorithm (TS) is used as a mutation operator (fig. 4). TS may be seen as a kind of mutation directed towards better fitting individuals, while the standard mutation is direction-less. The CGA allows to locate promising areas of the research space.

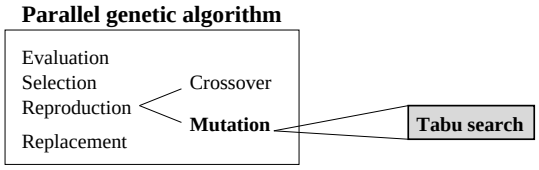

Figure 4: Parallel Synchronous Hybridization. For instance, a tabu search is used as a mutation operator in an CGA.

4.3 Parallel asynchronous hybridization

The parallel asynchronous hybridization (PAH) involves several search algorithms performing independently, and cooperating (exchange information) to find an optimum (fig. 5). We distinguish two different types of algorithms:

- **homogeneous** in which all the cooperating algorithms are the same
- **heterogeneous** in which different algorithms are used

Two kinds of cooperation may also be used:

- **global** in which all the algorithms search the same research space.
- **partial** in which case the problem to be solved is decomposed into sub-problems, each one having its own research space. Each algorithm is dedicated to the search of one of these spaces and yields a part of the solution. These parts are gathered to produce a solution.

The parallel asynchronous hybrid algorithm considered (heterogeneous global PAH), involves both the CGA and the MTS. The two algorithms execute concurrently and exchange individuals. Several issues should be addressed: which individuals should be exchanged? when? how are they integrated in the other algorithm?

In order to make the cooperation as efficient as possible, care should be taken to avoid exchanging several times the same individual. Hence, the diversity of exchanged solutions should be maximized. This requirement is difficult to fulfill. According to this last point, a PAH where the genetic algorithm no longer solves the QAP but aims at maximizing the diversity between the individuals of its population while the MTS optimizes the points that are yielded by the CGA conveys much attraction. The CGA acts like a diversification mechanism for MTS. The CGA generates and selects starting points for MTS.

A good initial population for tabu search should have high diversity. We must define a fitness function for CGA as the maximum "distance" between an individual and the tabu search actual population. To quantify this distance, we may use the hamming distance or a more sophisticated one which is the number of pair exchanges needed to equalize two permutations.

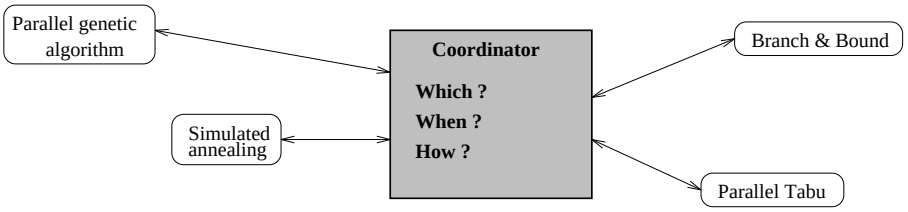

Figure 5: Parallel Asynchronous Heterogeneous Hybridization. Several search algorithms cooperate, co-adapt, and co-evolve a solution.

5 Performance evaluation

Our study aims at comparing the efficiency of parallel hybrid methods for solving the QAP. We are using the QAPLIB benchmark suite [4]. Five test instances, different in size and structure, have been selected: NUG20, NUG30 [17], STE36a [24], ELS19 [6] and SKO64 [23]. Table 1 summarizes the characteristics of the different test instances.

Instance	Size	Nature	Flow dominance
ELS19	19	practical application	very high
NUG20	20	randomly generated	low
NUG30	30	randomly generated	low
STE36a	36	practical application	high
SKO64	64	randomly generated	low

Table 1: Test suite problem characteristics.

A comparison with other meta-heuristics taken from the literature has been done:

- **An evolution strategy** $(1, \lambda)$-ES (CES) has been experimented in [16]. The distinction between an ES and a GA is that, the first models evolution by asexual reproduction with mutation and selection. In addition to these operators, the GA uses recombination (crossover) of different individuals. For large instances (SKO64), a sequential hybridization with HC has been used.

- **Simulated annealing**: The main applications of simulated annealing to the QAP have been proposed in [3][14][28]. The most recent works are presented in [10][11]. Results of simulated annealing algorithm (SA) are taken from [10]. A sequential hybridization (HSA) which combines SA and a tailored QAP heuristic was proposed in [11].

- **Tabu search** (TS): The first use of tabu search to solve the QAP was made by Skorin-Kapov [23]. We report here the results of Taillard's work, where the size of the tabu list was varying dynamically [25]. The tabu list prohibits moves where both swapped elements would be mapped to locations they had occupied in the last iterations. The CPU-time per iteration is given by $6.2n^2\mu s$.

Each heuristic was run from 5 to 30 times to obtain an average performance estimate. Our performance results are based on a C/PVM implementation on a 16-node DEC Alpha processor farm. The tables below show the best, worst, average value and the standard deviation of the solutions obtained for the chosen instances. The search cost was estimated by the number of function evaluation, and the CPU-time. It is difficult to make a comparison which is based on the CPU-time, because the different heuristics have been implemented on different architectures (SA and HSA on a VAX 6420, TS on a 10-node T800, and CES on a IBM RS6000).

According to the results, we observe that the MTS is well fitted to instances with low flow dominance but its performance decreases in other cases. Unlike the MTS, the CGA performs well on instances with high flow dominace. This behavior conforts our intuition of landscapes of the different instances. High flow

Table 2: Results for Elshafei's instance ELS19. Best known solution=17212548.

Heuristic	Runs	Best	Worst	Average	Standard Deviation	Average Evaluations	Average CPU(s)
HC	10	17 937 024	25 463 274	21 798 726	2 571 051.4	3593.8	0.4
TS (1000 it)	30	-	-	17 780 562	-	171 000	-
CES (200 gn)	10	17 212 548	23 247 770	19 218 337	2 094 638.9	10 486	1.9
CGA (100 gn)	10	17 212 548	17 937 024	17 379 040	284 712.44	33 280	38
MTS (100 it)	10	17 212 548	18 057 498	17 651 183	352 413.72	273 600	3
SH (100 gn + 100 it)	10	17 212 548	17 937 024	17 301 154	217 362.58	50 380	38
PSH (5 * (10 gn + 3 it))	10	17 212 548	21 112 254	19 892 646	1 697 006	673 280	27

Table 3: Results for Nugent's instance NUG20. Best known solution=2570.

Heuristic	Runs	Best	Worst	Average	Standard Deviation	Average Evaluations	Average CPU(s)
HC	10	2604	2724	2671	35.90	2001.6	0.2
SA	5	2570	2614	2591	15.93	-	43.9
TS (1000 it)	30	-	-	2573	-	190 000	-
CES (1000 gn)	10	2570	2610	2592	13.22	53 276	10.5
CGA (200 gn)	10	2604	2670	2643	22.84	66 560	56
MTS (100 it)	10	2570	2588	2573	5.22	304 000	3
HSA	5	2570	2588	2574	7.2	-	177.3
SH (200 gn + 100 it)	10	2570	2602	2589	10.74	85 560	59
PSH (3 * (20 gn + 10 it))	10	2570	2574	2571	1.83	1 479 168	42

dominance implies, for the landscape associated to the instance, a collection of rugged plateaus at different levels of fitness; low flow dominance corresponds to a rugged landscape without any dominating peak.

Consequently, it is expected that the PSH performs better than both CGA and MTS on instances with medium flow dominance as long as their landscape is intermediate and the PSH combines the advantages of this two algorithms.

6 Conclusion and future work

Modern meta-heuristics have been used to solve a NP-complete problem: the quadratic assignement problem. A comparative study with the best known heuristics has been carried out. The criteria of performances considered are the quality of the obtained solutions, the portion of the search space that is explored, and the amount of search time used for several standard instances of the QAP.

The experimental results show that each meta-heuristic explores and exploits the search space in its own way. No heuristic can be better than any heuristic on a wide spectrum of problems. To build more powerful search methods, combination of heuristics has been investigated.

Thus, parallel hybrid algorithms have been proposed. The use of parallel hybrid models is not restricted to the use of the "brute-force" of parallel computers. Rather, we think that a real cooperation of several

Table 4: Results for Nugent's instance NUG30. Best known solution=6124.

Heuristic	Runs	Best	Worst	Average	Standard Deviation	Average Evaluations	Average CPU(s)
HC	10	6230	6448	6322	66.93	10 322	2.2
SA	5	6136	6458	6216	121.8	-	162.1
TS (1000 it)	30	-	-	6142	-	435 000	-
CES (10 000 gn)	10	6124	6150	6135	9.31	1 050 181	372.4
CGA (300 gn)	10	6236	6402	6315	52.09	99 840	76
MTS (200 it)	10	6128	6178	6155	15.62	1 392 000	33
HSA	5	6124	6158	6138	11.13	-	633.1
SH (300 gn + 200 it)	10	6174	6280	6216	38	186 840	113
PSH (3 * (30 gn + 20 it))	10	6128	6192	6166	23.1	6 711 552	122

Table 5: Results for Steinberg's instance STE36a. Best known solution=9526.

Heuristic	Runs	Best	Worst	Average	Standard Deviation	Average Evaluations	Average CPU(s)
HC	10	10 188	11 122	10 447	265.18	17 389	5.2
TS (1000 it)	30	-	-	9917	-	630 000	-
CES (10 000 gn)	10	9564	9808	9701.2	87.92	1 032 511	489.2
CGA (350 gn)	10	10002	11022	10 485	324.96	116480	88
MTS (100 it)	10	9718	9964	9886	71.78	1 008 000	36
SH (350 gn + 100 it)	10	9748	10280	9999.2	158.85	179 480	106
PSH (10 * (35 gn + 10 it))	10	9606	10782	10097	299.31	16 244 480	392

Table 6: Results for Skorin-Kapov's instance SKO64. Best known solution=48498.

Heuristic	Runs	Best	Worst	Average	Standard Deviation	Average Evaluations	Average CPU(s)
HC	10	49082	49 994	49 524.0	321.27	188 649	241.6
SA	5	-	-	48 794.4	-	-	2806.7
TS (1000 it)	30	-	-	48 837.5	-	2 016 000	-
CES (50 000 gn)	10	48 752	48 924	48 815.2	52.04	5 045 471	9640.0
CGA (600 gn)	10	49 546	50 056	49 859	177.08	199 680	150
MTS (1000 it)	10	48 668	49 148	48 934	139.86	3.226 e+7	1789
HSA	5	-	-	48 815.2	-	-	2602.9
CES (50 000 gn) + HC	10	48 566	48 814	48 697.6	71.70	5 058 082	9656.6
SH (600 gn + 250 it)	10	49 402	49 772	49 339.4	204.28	703 680	595
PSH (3 * (60 gn + 25 it))	5	48 690	49 126	48 981.2	156.67	3.877 e+7	2244

search methods will lead to better solutions, by the way of a co-evolution of partial solutions, or different search methods, or a combination of both.

References

[1] D. E. Brown, C. L. Huntley, and A. R. Spillane. A parallel genetic heuristic for the quadratic assignment problem. In *Third Int. Conf. on Genetic Algorithms ICGA'89*, pages 406–415. Morgan Kauffmann, San Mateo, USA, July 1989.

[2] E. Buffa, G. Armour, and T. Vollmann. Allocating facilities with CRAFT. *Harvard Business Review*, 42:136–158, 1964.

[3] R. Burkard and F. Rendl. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 17:169–174, 1984.

[4] R. E. Burkard, S.Karisch, and F. Rendl. Qaplib: A quadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 1991.

[5] T. D. Crainic, M. Toulouse, and M. Gendreau. *Towards a taxonomy of parallel tabu search algorithms.* Technical Report CRT-933, Centre de Recherche sur les Transports, Université de Montreal, Sep 1993.

[6] A. Elshafei. Hospital layout as a quadratic assignment problem. *Operational Research Quarterly*, 28(1):167–179, 1977.

[7] M. Garey and D. Johnson. *Computers and Intractability: A guide to the theory on NP-completeness.* W.H. Freeman and Co. Publishers, New York, 1979.

[8] F. Glover. Tabu search - part I. *ORSA Journal of Computing*, 1(3):190–206, 1989.

[9] J. J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In L. Davis, editor, *Genetic algorithms and Simulated annealing*, Research Notes in Artificial Intelligence, pages 42–60, San Mateo, CA, USA, 1987. Morgan Kaufmann.

[10] S. S. Heragu and A. S. Alfa. Experimental analysis of simulated annealing based algorithms for the layout problem. *EJORDT*, 57(2):190–202, 1992.

[11] S. S. Heragu and A. Kusiak. Efficient models for the facility layout problem. *European Journal of Operational Research*, 53:1–13, 1991.

[12] J. H. Holland. *Adaptation in natural and artificial systems.* Michigan Press University, Ann Arbor, MI, 1975.

[13] T. C. Koopmans and M. J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[14] J. L. Lutton and E. Bonomi. The asymptotic behaviour of quadratic assignment problems: A statistical mechanics approach. *European Journal of Operational Research*, 26:295–300, 1986.

[15] H. Muhlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In J. Schaffer, editor, *Third Int. Conf. on Genetic Algorithms ICGA'89*, pages 416–421, 1989.

[16] V. Nissen. Solving the quadratic assignment problem with clues from nature. *IEEE Transactions on Neural Networks*, 5(1):66–72, Jan 1994.

[17] C. Nugent, T. Vollmann, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16:150–173, 1968.

[18] P. M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 16:1–42, 1994.

[19] P. Preux and E. G. Talbi. Assessing the evolutionary algorithm paradigm to solve hard problems. *CP95 Workshop on Studying and Solving Really Hard Problems, Marseille*, 1995.

[20] C. R. Reeves. *Modern heuristic techniques for combinatorial problems.* Blackwell Scientific Publication, 1993.

[21] W. T. Rhee. A note on asymptotic properties of the quadratic assignment problem. *Operations Research Letters*, 7:197–200, 1989.

[22] S. Sahni and T. Gonzales. P-complete approximation problems. *Journal of the ACM*, 23:556–565, 1976.

[23] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2:33–45, 1990.

[24] L. Steinberg. The backboard wiring problem: A placement algorithm. *SIAM Review*, 3:37–50, 1961.

[25] E. Taillard. Robust tabu search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[26] E. G. Talbi. *Parallélisme et Applications Irrégulières*, chapter Algorithmes génétiques parallèles : Techniques et application, pages 29–48. Hermes, 1995.

[27] T. E. Vollmann and E. S. Buffa. The facility layout problem in perspective. *Management Science*, 12(10):450–4468, 1966.

[28] M. R. Wilhelm and T. L. Ward. Solving quadratic assignment problems by simulated annealing. *IIE Transactions*, 19(1):107–119, 1987.

[29] D. H. Wolpert and W. G. Macready. *No Free Lunch Theorems for Search.* Technical Report SFI-TR-95-02-010, The Santa Fe Institute, Santa Fe, NM, USA, Feb 1995.