A GA-ACO-Local Search Hybrid Algorithm for Solving Quadratic Assignment Problem

Yi-Liang Xu, Meng-Hiot Lim, Yew-Soon Ong, Jing Tang
Intelligent System Center, Research Techno Plaza
Nanyang Technological University, Singapore 637553
{pg04266316, emhlim, asysong, pg04159923}@ntu.edu.sg

ABSTRACT
In recent decades, many meta-heuristics, including genetic algorithm (GA), ant colony optimization (ACO) and various local search (LS) procedures have been developed for solving a variety of NP-hard combinatorial optimization problems. Depending on the complexity of the optimization problem, a meta-heuristic method that may have proven to be successful in the past might not work as well. Hence it is becoming a common practice to hybridize meta-heuristics and local heuristics with the aim of improving the overall performance. In this paper, we propose a novel adaptive GA-ACO-LS hybrid algorithm for solving quadratic assignment problem (QAP). Empirical study on a diverse set of QAP benchmark problems shows that the proposed adaptive GA-ACO-LS converges to good solutions efficiently. The results obtained were compared to the recent state-of-the-art algorithm for QAP, and our algorithm showed obvious improvement.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search − Heuristic methods; G.1.6 [**Mathematics of Computing**]: Optimization − Global optimization.

General Terms
Algorithms, Performance, Experimentation

Keywords
Genetic Algorithm, Ant Colony Optimization, Local Search, Hybrid Algorithm, Adaptive Parameter Control.

1. INTRODUCTION
The quadratic assignment problem (QAP), which was first introduced by Koopmans and Beckmann [6] to solve a facility location problem, is a class of NP-hard problems. It can be described as follows. A problem of size n, can be represented by two ($n\times n$) matrices $\boldsymbol{D} = [d_{ij}]$ and $\boldsymbol{F} = [f_{ij}]$ ($1\le i,j\le n$). Matrix $\boldsymbol{D}$ can be interpreted as a distance matrix, i.e. d_{ij} represents the distance between location i and j, and $\boldsymbol{F}$ is normally referred to as the flow matrix, i.e. f_{ij} denotes the flow of materials from facility i to j. The goal is to find a permutation π of the set $\boldsymbol{M} = \{1, 2, 3,\ldots, n\}$, to minimize the objective function $C(\pi)$ described by Eq. (1):

$$C(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}F_{\pi_i\pi_j} \quad (1)$$

where π_i and π_j denote the i-th and the j-th element respectively in the permutation π.

There are many applications that can be modeled as QAP. Besides the facility location problem mentioned above, other applications of QAP include backboard wiring (Steinberg [7]), campus planning (Dickey and Hopkins [8]), ranking of archaeological data (Krarup and Pruzan [9]) and hospital layout planning (Elshafei [10]) and so on. The objective or cost function of these problems can be written in the general form of Eq. (1) or other equivalent variations.

The QAP has been a topic of frequent study in combinatorial optimization. The QAP is NP-hard, and only enumerative approaches are known to solve them optimally. However, the computational time cost will increase exponentially with respect to n and thus become unmanageable. Generally, it is considered unsolvable for QAP with size larger than 25 through exhaustive methods. But in practice, many applications are significantly larger than 25. Inevitably, heuristic approaches have played important role in algorithms capable of providing good solutions in reasonable time. In the last decades, much research has been devoted into the development of approaches such as greedy randomized search [12], simulated annealing [13,14,15], neural networks [16], tabu search [17,18], iterated local search [1], genetic algorithms (GA) [2,3,4,5,11,19,20,21], ant colony optimization (ACO) algorithms [23,24,25] and most recently, the hybridization of these algorithms, which are often referred to as memetic algorithms [26,27,28,29].

The focus of our work in this paper is a novel GA-ACO-LS hybrid algorithm for solving the QAP. It considers the synergy between genetic algorithm, ant colony optimization and local search to achieve synergistic balance between exploitative and explorative mechanisms of all three algorithms adaptively.

The rest of the paper is organized as follows. Section 2 introduces the meta-heuristics for QAP that forms our hybrid algorithm, which include genetic algorithm, ant colony optimization and local search. Section 3 presents the hybridization of the three meta-heuristic algorithms and the adaptive parameter control of the hybrid algorithm. The simulation results on a diverse set of QAP benchmark problems

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-186-4/06/0007…$5.00.

and the comparison with another recent hybrid algorithm for QAP are presented in Section 4. Finally, Section 5 concludes this paper.

2. META-HEURISTICS FOR QAP

We propose a hybrid algorithm incorporating genetic algorithm (GA), ant colony optimization (ACO) and local search for solving QAP. The three meta-heuristic algorithms for QAP in our implementation are briefly introduced below.

2.1 Genetic Algorithm

A genetic algorithm refers to a computational procedure that exhibits some form of exploitative behavior through selection, and explorative behavior through recombination and mutation. Genetic algorithm goes through many generations of explorative and exploitative search until it converges to a near optimal solution. We briefly introduce the genetic algorithm for QAP used in our hybrid algorithm.

```
procedure genetic algorithm for QAP
    P ← GenerateRandomPopulation()
    repeat
        (p1, p2) ← SelectParentPair(P)
        s ← Crossover(p1, p2)
        s ← Mutation(s, Pm)
        P ← PopulationUpdate(P, s)
    until termination condition met
end genetic algorithm for QAP
```

Figure 1. Genetic Algorithm for QAP

First, a population of solutions P is generated randomly. Then, within each evolutionary cycle, two candidate solutions designated as parents are selected. The two parents undergo crossover to produce an offspring solution s. After that, s mutates with a probability of mutation rate P_m. Finally, the resulting s, together with the existing population P, constitutes a new population for the next evolutionary cycle. This process is repeated, until the termination condition is met. The genetic operations involved are briefly described as follows.

In our genetic algorithm, tournament selection is used as the parents selection procedure. In this mechanism, two individuals are randomly selected from the reproduction pool. The better individual is designated as one parent. In this way, two parents are selected and the better parent is designated as the first parent for reproducing one offspring by crossover.

Crossover is used in GA to inherit constructive information from parents throughout the generations. From the description of QAP above, the absolute position of each allele, i.e. the assignment of facility $\pi(i)$ to location i carries the constructive information of QAP. Therefore we adopted uniform crossover in our algorithm since it is good at preserving the absolute position information of alleles.

Mutation serves as the secondary search operator of GA for exploring new search regions by altering a certain number of genes of a chromosome randomly. We use a scramble mutation in our implementation whereby $n/2$ genes in an individual are randomly picked and swapped.

The population update mechanism used in our implementation is similar to the well-known steady-state reproduction. Only one solution instead of an entire population is generated in each generation. If the new solution is better than the worst existing solution in the population, it is accepted to replace the worst member to form a new population, otherwise it is ignored. Besides this rule, in our implementation, the new solution is also ignored if an equivalent solution is already in the existing population to maintain the diversity of the population.

2.2 Ant Colony Optimization

Ant colony optimization (ACO) is a class of constructive meta-heuristic algorithms. It imitates the behaviors of real ants of a colony when foraging for food. Each artificial ant agent constructs a solution based on the constructive information, which is termed pheromone, provided by previous ant agents that have already built solutions. After having built new solutions, the artificial ants update the pheromone traces, taking into account the quality of the existing solutions. We briefly introduce the ACO algorithm for QAP used in our hybrid algorithm.

Figure 2 shows the general framework of the ant colony optimization for QAP in our implementation. First, a colony/population P of artificial ants which represent feasible solutions of QAP is initialized randomly and the pheromone traces Ph are initialized. After that, in the evolution loop, the pheromone traces Ph are updated based on the colony members. Based on the updated pheromone traces, the ACO solution constructor constructs a feasible solution s for QAP, which is designated as the new generated solution of the iteration. The new generated solution s, together with the existing colony members, constitutes a new colony/population of ants (solutions) following the same rules in the population update procedure in GA as described above. This process is repeated until the termination condition is met. The ACO operations involved are briefly described as follows.

```
procedure ant colony optimization for QAP
    P ← GenerateRandomPopulation()
    Ph ← PheromoneInitialization()
    repeat
        Ph ← PheromoneUpdate(P)
        s ← SolutionConstruction(Ph)
        P ← PopulationUpdate(P, s)
    until termination condition met
end ant colony optimization for QAP
```

Figure 2. Ant Colony Optimization for QAP

Note that the assignment of facility $\pi(i)$ to location i carries the constructive information for QAP. Therefore, we define the pheromone trace $\tau_{ij}(t)$ $(1 \le i,j \le n)$ as the preference of assigning facility j to location i in iteration t.

With this definition of pheromone information, an infeasible QAP solution can be constructed as follows. In iteration t, given a

location i, an ant decides to assign facility j to this location with the following probability:

$$p_{ij}(t) = \frac{\tau_{ij}}{\sum_{l=1}^{n} \tau_{il}} \quad (2)$$

where l indicates facility l that has not been assigned. An ant selects the locations in a random order and determines the corresponding facilities by adopting Eq. (2) iteratively. This way, a feasible solution for QAP is constructed based on the pheromone traces.

At the beginning of each iteration t, the pheromone traces are updated from the last iteration and then used for the construction of solution in the current iteration:

$$\tau_{ij}(t) = \rho \cdot \tau_{ij}(t-1) + \sum_{k=1}^{P} \Delta\tau_{ij}^{k} \quad (3)$$

where ρ, with $0<\rho<1$, is the persistence of the pheromone traces, P is the colony (population) size, and $\Delta\tau_{ij}^{k}$ is the amount of pheromone ant k ant puts on the assignment coupling (i,j). It is calculated by:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/J^{k} & \text{if facility } j \text{ is assigned to location } i \text{ in ant } k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where J^k is the objective value of ant k.

To avoid premature convergence caused by few dominant pheromone traces, we borrow the procedure from the MAX-MIN ant system [25] and define the allowed range of the pheromone traces $[\tau_{min}, \tau_{max}]$. After the pheromone update, if we have any $\tau_{ij} > \tau_{max}$, we set $\tau_{ij} = \tau_{max}$; similarly, if $\tau_{ij} < \tau_{min}$, we set $\tau_{ij} = \tau_{min}$.

Form Eq.(3), we can deduce that the possible upper bound of any pheromone trace value is:

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{J^{opt}} \cdot P \quad (5)$$

where J^{opt} is the optimal solution value. Therefore, we initialize all the pheromone trace values to τ_{max}. We set τ_{min} as follows:

$$\tau_{min} = \tau_{max} / n \quad (6)$$

2.3 Local Search

In our work, we adopted a simple tabu search to locally optimize each new generated solution. It is a simplified version of the well-known robust tabu search [18] for QAP. It iteratively searches the neighborhood of the current solution and uses the tabu list and tabu rules to avoid being stuck in local optima. The search mechanisms are briefly described next.

Let π be a starting solution and $N(\pi)$ the corresponding set of neighboring solutions, which can be derived from π by exchanging two different facilities, π_r and π_s. Create a facility tabu list of size $n/5$ and clear the tabu list. The local search repeats the following steps:

1) Checks all solutions in $N(\pi)$ of the current solution.
2) If the best solution in $N(\pi)$ is better than the current solution, update the current solution with the best solution in $N(\pi)$ and clear the tabu list.
3) If the best solution in $N(\pi)$ is no better than the current solution, update the current solution with the best solution in $N(\pi)$ only if the two exchanged facilities are not in the tabu list. Add the two exchanged facilities into the tabu list maintained as a FIFO queue of size $n/5$.
4) Go to Step 1).

These steps are repeated until the maximum number of iterations I_{max} has been executed. The cost of exchanging π_r and π_s can be efficiently calculated in constant time using the shortcut scheme in [18].

3. GA-ACO-LS HYBRID ALGORITHM

This paper proposes a novel hybrid algorithm as a synergy between the genetic algorithm, ant colony optimization and local search previously described in section 2. The purpose is to promote cooperation and competition among the different heuristic algorithms, working together to accomplish the shared optimization goal. In the process, we adopted an adaptive parameter control mechanism to balance the explorative and exploitative behaviors of these three heuristic procedures as the search progresses.

3.1 Flow of the Algorithm

The complete flow of the proposed hybrid algorithm is presented in Figure 3. First, a population/colony P of solutions (chromosomes/ants) was randomly initialized. A pheromone trace set Ph is also initialized as described above in Subsection 2.2. A parameter P_g is initialized to 0.5. In the evolution loop, either a genetic algorithm or an ant colony optimization procedure will be selected to generate a new solution. This choice is controlled by a Boolean function *bApplyGA* with the parameter P_g.

- If a GA procedure is chosen, it first selects two parents (p1, p2) from the existing population P as described above in Subsection 2.1. Through crossover and mutation, a new solution s is produced.
- If an ACO procedure is chosen, the pheromone traces Ph are updated by the population as described above in Subsection 2.2. Then the ACO solution construction procedure constructs a new solution s based on the updated pheromone traces as described above.

The resulting solution s, either by GA or ACO, is then improved by the local search procedure described in Subsection 2.3.

The final new solution s is added into the existing population P or discarded through the population update mechanism as described above in Subsection 2.1. The parameter P_g is updated accordingly as described in the next subsection. These steps are repeated until the termination condition is met. In our implementation, we terminate the algorithm when 20 times the population size number of new solutions are generated in succession and none of them is able to update the best uncovered solution.

```
procedure GA-ACO-Local Search
    P ← GenerateRandomPopulation()
    Ph ← PheromoneInitialization()
    Pg ← 0.5
    repeat
        if (bApplyGA(Pg) == TRUE)
            (p1, p2) ← SelectParentPair(P)
            s ← Crossover(p1, p2)
            s ← Mutation(s, Pm)
        otherwise
            Ph ← PheromoneUpdate(P)
            s ← SolutionConstruction(Ph)
        s ← LocalSearch(s)
        P ← PopulationUpdate(P, s)
        Pg ← UpdatePg()
    until termination condition met
end GA-ACO-Local Search
```

Figure 3. GA-ACO-LS Hybrid Algorithm

Both the genetic algorithm and ant colony optimization operate on a shared population. This serves to enhance the effort of accomplishing the shared optimization goal through population and pheromone updates of the hybrid algorithm. As shown in Figure 4, in one hand, GA keeps producing new solutions that replace the population members and improves the average quality of the population members. This improvement leads to the update on the pheromone traces, which enhances the ACO solution construction. On the other hand, ACO keeps constructing new solutions, which replace the worst solutions in the population and improves the average quality of the parent population members. This improvement enhances the GA convergence and increases the chance of further producing better solutions for GA. We believe this "win-win" collaborative pattern increases the chance of uncovering good solutions for the entire hybrid algorithm.

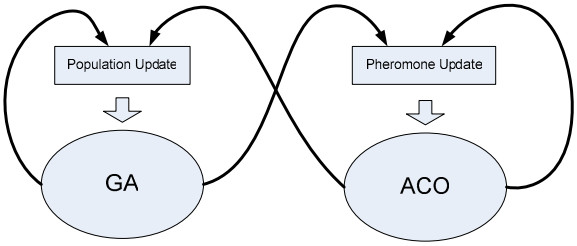

Figure 4. Collaborative hybridization

3.2 Adaptive Control for P_g

This collaborative hybridization pattern raises an issue on how to determine GA and ACO's weight in order to achieve an appropriate balance in explorative and exploitative behaviors. As described before, this is controlled by the parameter P_g. In our work, we adopted an adaptive control over the parameter P_g:

$$P_g = F(T) = \int_{-\infty}^{T} f(x)dx \tag{9}$$

where $f(x)$ is the probability density function (PDF) of *normal distribution*:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{10}$$

where σ is the standard deviation and μ is the mathematical expectation of *normal distribution*.

Variable T is initialized to μ, and P_g is thus initialized to 0.5. This means GA and ACO are initially awarded the equal chance to perform. At the end of each iteration, as described above in Subsection 2.1, if the new generated solution is better than the worst member of the existing member and is not in the existing population, it replaces the worst member of the population. The parameter $P_g=F(T)$ is then updated as follows:

- If the new solution is generated by GA operations, i.e. GA contributes to the entire algorithm, $T=T+\sigma/500$;
- If the new solution is generated by ACO operations, i.e. ACO contributes to the entire algorithm, $T=T-\sigma/500$;

where σ is the standard deviation. In other words, this simple mechanism awards the contributing algorithm with a higher chance to participate in the search. It is worth noting that the choice of the values for σ and μ would not affect the updated value of P_g. For *standard normal distribution*, we simply set σ = 1 and μ = 0.

This adaptive parameter control further introduces competition between GA and ACO besides the collaboration mentioned above. We contend that a good balance between these two features lead to good performance of the entire algorithm.

4. SIMULATION RESULTS

We carried out experiments on a diverse set of QAP benchmark problems, with n ranging from 40 to 100, from QAPLIB [22] to study the proposed algorithm. The programs were coded in C++ programming language and the computing resource was a DELL Pentium IV 2.8GHz computer. For each problem, 20 simulation trials were carried out and the statistical results are reported. Some parameters are set as follows after rough tuning:

Population Size = 100,

Mutation Rate P_m = 0.1,

# of Tabu Search Iteration $I_{max} = 8n$,

Persistence Factor ρ = 0.8.

To present the simulation results, we define several attributes to evaluate and compare the algorithms in terms of time and solution quality.

BK denotes the cost value of the best-known solution for a certain problem. *Time* is the average computation time in seconds upon termination of the algorithm. It is used to measure the speed of the algorithms in real computational time. *Avg_gap* is the

difference between the average objective value of the solutions obtained for all the algorithm runs and the best-known value of the objective function. *Rs* refers to the ratio of number of trials the algorithm finds the best-known solution to the number of all algorithm trials.

4.1 Fixed Versus Adaptive P_g

We first compare algorithms with different settings of P_g:

- P_g = 0.0, which essentially indicates a pure ACO-LS hybrid algorithm;
- P_g = 1.0, which essentially indicates a pure GA-LS hybrid algorithm;
- P_g = 0.5, in which GA and ACO are awarded equal chance to perform;
- P_g under adaptive control as described in Subsection 3.2.

Table 1 summarizes the simulation results of these 4 algorithms. From the table, the hybrid algorithm with adaptive P_g control appears to be the best, as evidenced by the fact that it produces the best or equals the best average solution quality in 14 out of 16 instances yet its computational time shows no significant difference compared to the other algorithms.

Table 1. Compare Different Settings of P_g

problem	n	BK	Pg =0.0			Pg =1.0			Pg =0.5			Adaptive Pg		
			Avg_gap	Time (s)	Rs	Avg_gap	Time (s)	Rs	Avg_gap	Time (s)	Rs	Avg_gap	Time (s)	Rs
tai40a	40	3139370	0.4268%	86.8	0%	0.3332%	87.8	0%	0.3816%	89.6	0%	**0.3282%**	88.4	0%
tai50a	50	4941410	0.6052%	172.7	0%	0.5438%	181.3	0%	0.5911%	167.3	0%	**0.5330%**	174.7	0%
tai60a	60	7205962	0.6374%	342.3	0%	**0.5957%**	353.3	0%	0.6351%	346.3	0%	0.5977%	338.3	0%
tai80a	80	13540420	0.5386%	782.8	0%	0.4202%	808.2	0%	0.4347%	854.3	0%	**0.4156%**	833.0	0%
tai100a	100	21123042	0.6851%	1424.3	0%	0.6035%	1470.0	0%	0.6041%	1476.4	0%	**0.5813%**	1543.9	0%
sko72	72	66256	0.0179%	556.3	40%	**0.0085%**	530.3	70%	0.0101%	556.1	50%	**0.0085%**	530.8	75%
sko80	80	13557864	0.0267%	760.9	10%	0.0197%	729.7	30%	**0.0179%**	723.7	20%	**0.0179%**	747.0	20%
sko90	90	115534	0.0069%	1011.1	25%	0.0059%	1068.4	35%	0.0069%	997.0	25%	**0.0048%**	1043.8	45%
sko100a	100	152002	0.0358%	1500.9	10%	0.0187%	1439.7	20%	0.0198%	1562.1	35%	**0.0173%**	1495.9	35%
sko100b	100	153890	0.0056%	1481.6	40%	0.0038%	1527.1	60%	0.0049%	1464.6	60%	**0.0026%**	1534.6	75%
sko100c	100	147862	0.0087%	1437.8	35%	**0.0032%**	1416.5	80%	0.0040%	1456.5	70%	0.0034%	1486.3	75%
sko100d	100	149576	0.0239%	1569.1	10%	0.0216%	1511.6	40%	0.0211%	1535.4	25%	**0.0184%**	1489.4	35%
sko100e	100	149150	0.0028%	1538.2	40%	**0.0000%**	1484.4	100%	0.0018%	1447.5	70%	**0.0000%**	1496.7	100%
sko100f	100	149036	0.0375%	1464.5	5%	**0.0204%**	1556.6	30%	0.0241%	1605.4	20%	**0.0204%**	1567.5	30%
wil100	100	273038	0.0055%	1555.2	10%	0.0039%	1502.0	40%	0.0034%	1453.8	35%	**0.0030%**	1486.4	35%
tho150	150	8133398	0.0793%	2764.8	0%	0.0733%	2943.5	0%	0.0777%	2936.3	0%	**0.0560%**	2869.4	0%

We further investigate how the parameter P_g changes as the algorithm runs. Figure 5 plots based on the number of solutions evaluated in a typical run on the benchmark problem tai40a.

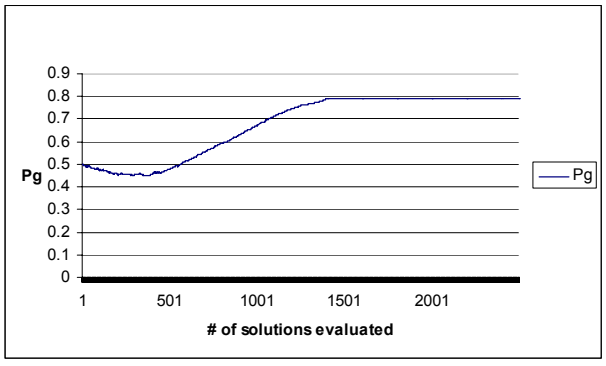

Figure 5. Adaptive control of P_g

In the plot, P_g first drops to below 0.5, which means ACO contributes more than GA. It dominates in the early stages of the algorithm. As the algorithm progresses, P_g increases and exceeds 0.5. In the final stage of the search, GA becomes the dominating search engine until P_g finally is maintained at a high level, which indicates that the search has converged.

This observation can be explained by the fact that ACO performs more exploitative behaviors than GA, while GA is relatively more explorative. In the early stages of the algorithm, the population is randomly initialized and thus quite diverse. Therefore, ACO's exploitative capacity of learning the best found solutions' feature and constructing solutions following these good solutions should be more productive than GA's explorative behaviors like crossover and mutation. However, as the population matures, the average solution quality is higher. It becomes much more difficult for ACO to construct further better solutions based on the current best solutions, which are already very good. On the other hand, GA's explorative capacity becomes more useful in broadening the search regions and bringing disturbance into the search process to introduce the diversity to the population.

4.2 Comparison with Another Hybrid Algorithm

We compare our hybrid algorithm with one recently reported by Stützle [1], a hybrid of evolutionary strategy (ES) and iterated local search (ILS).

The experiments reported in [1] were carried out on a PC with a 1.3 GHz CPU. Table 2 summarizes the comparison.

Table 2. Comparison with ES-ILS [1]

problem	*n*	Adaptive *Pg*			ES-ILS [1]		
		Avg_gap	*Time* (s)	*Rs*	*Avg_gap*	*Time* (s)	*Rs*
tai40a	40	0.3282%	88.4	0%	**0.28%**	1200	0%
tai50a	50	**0.5330%**	174.7	0%	0.61%	1200	0%
tai60a	60	**0.5977%**	338.3	0%	0.82%	1200	0%
tai80a	80	**0.4156%**	833.0	0%	0.62%	1200	0%
tai100a	100	**0.5813%**	1543.9	0%	0.69%	1200	0%
sko72	72	0.0085%	530.8	75%	**0.0012%**	1200	88%
sko81	80	0.0179%	747.0	20%	**0.0074%**	1200	52%
sko90	90	**0.0048%**	1043.8	45%	0.0057%	1200	40%
sko100a	100	0.0173%	1495.9	35%	**0.012%**	1200	30%
sko100b	100	**0.0026%**	1534.6	75%	0.0068%	1200	50%
sko100c	100	0.0034%	1486.3	75%	**0.0023%**	1200	60%
sko100d	100	**0.0184%**	1489.4	35%	0.021%	1200	0%
sko100e	100	**0.0000%**	1496.7	100%	0.0013%	1200	70%
sko100f	100	**0.0204%**	1567.5	30%	0.037%	1200	0%
wil100	100	**0.0030%**	1486.4	35%	0.0041%	1200	10%
tho150	150	**0.0560%**	2869.4	0%	0.068%	3600	0%

From the table, it is shown that our algorithm produces better average solutions in 11 out of 16 instances while produces poorer average solutions in only 5 instances. It is also observed that for instance sko100d and sko100f, the ES-ILS failed to uncover the best known solution, while our algorithm was able to achieve the success rate of 35% and 30% in 20 trials respectively. We attribute the superiority of our algorithm to the novel collaborative hybridization pattern of our approach.

5. CONCLUSIONS

In this paper, we proposed a GA-ACO-LS hybrid algorithm for solving QAP. We adopted a novel collaborative hybridization methodology, which is a combination of GA, ACO and local search. We further introduced an adaptive parameter control mechanism into the algorithms to balance the explorative and exploitative behaviors of the three heuristics at different stages of the algorithm. The simulation results were compared with another recently reported hybrid algorithm and we showed that our GA-ACO-LS hybrid algorithm produced better search performance.

6. REFERENCES

[1] Stützle, T. Iterated local search for the quadratic assignment problem, *European Journal of Operational Research*, in press, corrected proof, available online 13 May, 2005.

[2] Lim, M.H., Yuan, Y. and Omatu, S. Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. *Computational Optimization and Applications,* 2000;15:249-268.

[3] Lim, M.H., Yuan, Y. and Omatu, S. Extensive testing of a hybrid genetic algorithm for quadratic assignment problem. *Computational Optimization and Applications*, 2002;23:47-643.

[4] Drezner, Z. A new genetic algorithm for the Quadratic Assignment Problem. *INFORMS J. Comput., 15,* (2003), 320-330.

[5] Ahuja, R.K., Orlin, J.B., and Tiwari, A. A greedy genetic algorithm for the Quadratic Assignment Problem. *Computers and Operations Research, 27,* (2000), 917-934.

[6] Koopmans, T.C. and Beckmann, M.J. Assignment problems and the location of economic activities. *Econometrica, 25,* (1957), 53-76.

[7] Steinberg, L. The backboard wiring problem: a placement algorithm. *SIAM Rev, 3,* (1961), 37-50.

[8] Dickey, J.W. and Hopkins, J.W. Campus building arrangement using TOPAZ. *Transportation Rev, 6*, (1972), 59-68.

[9] Krarup, J. and Pruzan, P.M. Computer-aided layout design. *Math. Programming Study, 9,* (1978), 85-94.

[10] Elshafei, A.N. Hospital layout as quadratic assignment problem. *Operational Res. Quarterly, 28,* (1977), 167-179.

[11] Tate, D.M. and Smith, A.E. A genetic approach to the quadratic assignment problem. *Computer & Operations Research, 22,* 1 (1995), 73-83.

[12] Li, Y., Pardalos, P.M. and Resende, M. A greedy randomized adaptive search procedure for the quadratic assignment problem. In *Quadratic Assignment and Related Problems*. P. Pardalos and H. Wolkowicz (Eds.), AMS: Providence, Rhode Island, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1994;16:173-187.

[13] Burkard, R.E. and Rendl, F. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research, 17,* (1984), 169-174.

[14] Connolly, D.T. An improved annealing scheme for the QAP. *European Journal of Operational Research, 46,* (1990), 93-100.

[15] Laursen, P.S. Simulated annealing for the QAP—optimal tradeoff between simulation time and quality. *European Journal of Operational Research, 63,* (1993), 238-243.

[16] Chakrapani, J. and Skorin-Kapov, J. A connectionist approach to the quadratic assignment problem. *Computers & Operations Research, 19,* 13-14 (1992), 287-295.

[17] Skorin-Kapov, J. Extensions of a tabu search adaption to the quadratic assignment problem. *Computer & Operations Research, 21,* 8 (1994), 855-865.

[18] Taillard, E. Robust taboo search for the quadratic assignment problem. *Parallel Computing, 17,* (1991), 443-455.

[19] Brown, D.E., Huntley, C.L. and Spillane, A.R. A parallel genetic heuristic for the quadratic assignment problem. In *Proceedings of International Conference on Genetic Algorithms*. 1989, 406-415.

[20] Fleurent, C. and Ferland, J. Genetic hybrids for the quadratic assignment problem. In *Quadratic Assignment and Related Problems*. P. Pardalos and H. Wolkowicz (Eds.), AMS: Providence, Rhode Island, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1994;16:173-187.

[21] Merz, P. and Freislenben, B. A genetic local search approach to the quadratic assignment problem. In *Proceedings of International Conference on Genetic Algorithms (ICGA'97)*. Morgan Kaufmann: San Mateo, CA, 1997, 465-472.

[22] Burkard, R.E., Karisch, S.E. and Rendl, F. QAPLIB—A quadratic assignment problem library. *Journal of Global Optimization, 10*, (1997), 391-403.

[23] Maniezzo, V. and Colorni, A. The ant system applied to the Quadratic Assignment Problem. *IEEE Transaction on Knowledge and Data Engineering, 11,* 5, (1999), 769-778.

[24] Stützle, T. MAX-MIN ant system for Quadratic Assignment Problem. Technical Report AIDA-97-04, Intellectics Group, Department of Computer Science, Darmstadt University of Technology, Germnay, July 1997.

[25] Stützle, T. and Hoos, H. H. MAX-MIN ant system. *Future Generation Computer Systems*, *16,* 8, (2000), 889-914.

[26] Ong, Y. S., Lim, M. H., Zhu, N. and Wong, K. W. Classification of Adaptive Memetic Algorithms: A Comparative Study. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, *36,* 1, (2006).

[27] Ong, Y. S. and Keane, A. J. Meta-Lamarckian Learning in Memetic Algorithm. *IEEE Transactions On Evolutionary Computation*, *8*, 2, (2004), 99-110.

[28] Acan, A. GAACO: A GA + ACO Hybrid for Faster and Better Search Capability. In *Proceedings of Third International Workshop on Ant Algorithms (ANTS'02)*. Brussels, Belgium, 2002, 300-301.

[29] Reimann, M., Shtovba, S. and Nepomuceno, E. A hybrid ACO-GA approach to solve Vehicle Routing Problems, *Student Papers of the Complex Systems Summer School, Budapest,* Santa Fe Institute, (2001).