Ant Colony with Stochastic Local Search for the Quadratic Assignment Problem

Malek Mouhoub and Zhijie Wang
University of Regina
Wascana Parkway, Regina, SK, Canada, S4S 0A2
{mouhoubm,wang207z}@cs.uregina.ca*

Abstract

The existing Ant Colony Optimization (ACO) Algorithms for the Quadratic Assignment Problem (QAP) are often combined with two kinds of Stochastic Local Search (SLS) methods: the 2-opt local search and the tabu local search. In this paper, these two SLS methods are respectively improved according to the properties of ACO and QAP. For the 2-opt local search, a new random walk strategy is used to avoid a quick stagnation into local optima. Moreover, a forward-looking strategy is proposed to explore the neighborhood more thoroughly. In the case of tabu local search, a random walk strategy is also employed to avoid getting stuck at local optima. Experimental evaluation of the ACO algorithms combined with the improved local search proposed in this paper are conducted on problems from the well known QAPLIB library. The results demonstrate that each ACO algorithm, combined with its respective improved local search, has a better performance, in terms of the quality of the solution returned, than the ACO algorithm with the original local search techniques. Moreover, we also noticed that the improved methods outperform each other for different classes of problems.

1. Introduction

The Quadratic Assignment Problem (QAP) is an important theoretical and practical problem. Indeed, many real world applications such as backboard wiring, typewriter keyboard design, scheduling and many others can be formulated as QAPs. The QAP can be described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. The goal then is to find an assignment such that the sum of the products of the flows and distances is minimal. More formally, we are given \( n \) facilities and \( n \) locations, two \( n \times n \) matrices \( A = [a_{ij}] \) and \( B = [b_{ij}] \), where \( a_{ij} \) is the flow between facilities \( i \) and \( j \) and \( b_{rs} \) is the distance between locations \( r \) and \( s \). Given \( S(n) \) the set of all permutations of the set \( \{1, \ldots , n\} \), an objective function \( f \) of a feasible solution \( \phi \in S(n) \) is defined as follows:

\[
f(\phi) = \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} a_{\phi(i)\phi(j)}
\]

where \( \phi \) corresponds to the facility assigned to the location \( i \) in the current solution \( \phi \). An optimal solution to the QAP, noted \( \phi^* \), is a feasible solution satisfying the following:

\[
f(\phi^*) = \min_{\phi \in S(n)} f(\phi).
\]

The QAP is a NP-hard optimization problem. Generally, it is hard for the traditional exact algorithms to solve those QAPs whose scales (number of facilities | locations) are larger than 20. Therefore, to practically solve the QAP one has to use approximation algorithms that can find sub-optimal solutions in a short time such as Simulated Annealing Algorithm[1], Tabu Search[2], Genetic Algorithm[3] and Ant Colony Optimization (ACO) algorithm[4].

Among those approximation algorithms, ACO method is a new kind of evolutionary algorithm which was presented in the early 90s. More than ten years of research indicates that, ACO algorithm has a strong ability of finding the best solutions to combinatorial optimization problems and has the merits of distributed computation, easy combination with other algorithms and robustness. Variants of ACO algorithms for QAP include the Ant System[6] which is the first ACO algorithm for the QAP, the Max-Min Ant System (MMAS)[7] which improves the Ant System, and the Hybrid Ant System for the QAP[8].

Generally, ACO algorithms are always combined with SLS so that the local search has been implicitly a part of the ACO procedure[7, 9]. In this paper, two kinds of popular local search used in ACO method, 2-opt local search and tabu local search, are improved by a careful selection of the potential solution at each iteration of the algorithm. The experiments conducted on problems taken from the well known Quadratic Assignment Problem Library (QAPLIB)[5] show that the improved algorithms obviously have better performances than the original ones. Moreover, we also noticed that the improved methods outperform each other for different types of problems. As reported in [10], we believe that this is due to the “unstructureness” in the solution space of the random instances we used for our tests.

The rest of the paper is structured as follows. In the next
Section we introduce the MMAS for solving QAPs. We will then present in Section 3 the improvements we have made to the two local search techniques used in the ACO algorithm to solve the QAP. Section 4 is dedicated to the experimental evaluation we have conducted in order to evaluate the performance of our improved search techniques. Conclusion and future work are finally covered in Section 5.

2. MMAS for QAPs

Recent research shows that ants can lay a kind of pheromone on the paths they pass by during their foraging process. Such pheromone can guide the ants’ moving choices, i.e. ants will prefer the paths with high intensity of the pheromone trail. Therefore, the collective foraging behavior of a great deal of ants shows a phenomenon of information positive feedback. The shorter a path is, the more ants pass by and then the more intense pheromone is on this path, the higher probability for later ants to choose this path. The ant colony searches the shortest way from its nest to the food by such an information communication way. The ACO algorithm is actually an optimization algorithm simulating the natural colony’s foraging behavior. Specifically, in the ACO algorithm for the QAP, ants will lay the pheromone on the pair \((i, j)\) if they assign the facility \(j\) to the location \(i\) rather than on a path.

MMAS is a representative ACO algorithm proposed by Thomas Stützle[7]. Algorithm 1 describes the MMAS algorithm applied to the QAP problem. In the algorithm, \(\tau_{ij}(t)\) indicates the pheromone trail associated with assigning the facility \(j\) to the location \(i\) at time \(t\). Specifically, in MMAS the pheromone trail is limited in the area of \([\tau_{\min}, \tau_{\max}]\) to avoid the early stagnation into local optima. All the values exceeding the limitation will be set to \(\tau_{\min}\) or \(\tau_{\max}\) compulsively. \(\tau_{\min} = 2/est\_opt\), where \(est\_opt\) is the objective function value of a randomly generated solution improved by local search, \(\tau_{\max} = 5 \cdot \tau_{\min}\). Line 10 refers to 2-opt local search and tabu local search which are described respectively in algorithms 2 and 3. In both local search algorithms, the neighborhood \(N(\varphi)\) of a given solution \(\varphi\) is the set of solutions which can be obtained by exchanging two facilities. In the tabu local search, a neighboring solution that places facilities \(i\) and \(j\) on locations \(r\) and \(s\) is tabu, if in the past \(t_l\) iterations facilities \(i\) and \(j\) have been placed on locations \(r\) and \(s\) respectively. Here \(t_l\) is the tabu list length. The solution satisfying the aspiration criteria means a new best solution since the start of the search. The tabu list length \(t_l\) is chosen randomly during the search from the interval \([0.9 \cdot n, 1.1 \cdot n]\) where \(n\) is the number of facilities (a new random value is chosen every 2.2 \(\cdot\) \(n\) iterations). The maximum number of iterations of tabu search is set to 4 \(\cdot\) \(n\).

Algorithm 1
1: Compute \(\tau_{\max}\) and \(\tau_{\min}\).
2: Initialize the pheromone matrix with all \(\tau_{ij}\) equal to \(\tau_{\max}\).
3: Place \(m\) ants on the location 1.
4: for \(k = 1\) to \(m\) do
5: \hspace{1em} for \(i = 1\) to \(n\) do
6: \hspace{2em} Ant \(k\) randomly chooses a facility \(j\) from \(N(k)\) and then places it on location \(i\) with a probability given by: \(P_i^k(j) = \frac{\tau_{ij}^k(\varphi_{opt})}{\sum_{j \in N(i)} \tau_{ij}^k(\varphi_{opt})}\)
7: \hspace{1em} \(N(k)\) is the set of facilities that ant \(k\) hasn’t assigned yet. \(N(k)\) should be adjusted dynamically, i.e. facility \(j\) should be deleted from the \(N(k)\) at once.
8: \hspace{1em} Go back to Line 4.
9: \hspace{1em} for \(k = 1\) to \(m\) do
10: \hspace{2em} Improve the solution of ant \(k\) with 2-opt local search or tabu local search.
11: \hspace{1em} Go back to Line 4.
12: \hspace{1em} Update the pheromone matrix according to
\[
\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta \tau_{ij}^{best}
\]
where \(\rho\) is the pheromone volatilization coefficient and
\[
\Delta \tau_{ij}^{best} = \begin{cases} 
1/f(\varphi_{opt}) & \text{if facility } j \text{ is assigned on location } i \text{ in the solution } \varphi_{opt} \\
0 & \text{otherwise}
\end{cases}
\]
where \(\varphi_{opt}\) is the best solution found in the current cycle.
13: if \(time_{elapsed} < time_{max}\) then
14: \hspace{1em} \(N(k) = \{1, \ldots, n\}\), i.e. reset the ants’ sets of facilities that haven’t been assigned.
15: \hspace{1em} Go back to Line 4.
16: else
17: \hspace{1em} print the best solution found.
18: end if
end

Algorithm 2
1: Search the neighborhood of solution \(\varphi\), and then record the best solution \(\varphi'\) in the neighborhood.
2: if \(f(\varphi') < f(\varphi)\) then
3: \hspace{1em} \(\varphi = \varphi'\).
4: Go back to Line 1.
5: end if
end

Algorithm 3
1: Search the neighborhood of solution \(\varphi\), and then choose the best solution \(\varphi'\) among the non-tabu solutions and the solutions satisfying the aspiration criteria.
2: \(\varphi = \varphi'\).
3: Update the tabu lists.
4: if \(Iteration_{max} < Iteration_{max}\) then
5: \hspace{1em} Go back to Line 1.
6: end if
end
3. Improved Local Searches

3.1. Improved 2-opt local search

We propose in the following two improvements made to the 2-opt local search. In order to prevent the search from being trapped in a local optima, the first one concerns a better selection of the neighboring solution to consider in the next iteration. The second improvement allows the search algorithm to consider more neighbors when it is stuck in a local optima. In the following we will describe each of the two improvements.

3.1.1 Neighbor selection using random walk.

In the original 2-opt local search, the best potential solution in the current solution’s neighborhood is chosen as the initial solution in the next iteration. This will however lead the search quickly to local optima. To overcome this difficulty, we randomize the search algorithm such that the best solution is chosen with probability \( q_0 \) while a random solution is chosen with probability \( 1 - q_0 \) where \( q_0 \in [0, 1] \). More precisely the method works as follows. Choose a couple \((r, s)\), and then get the initial solution in the next iteration by exchanging the facilities on locations \( r \) and \( s \) in the current solution. The couple \((r, s)\) is chosen according to the formulation 1 below.

\[
(r, s) = \begin{cases} 
\arg \max_{(r,s) \in Q} -\Delta(\varphi, r, s) & \text{if } q < q_0 \\
(R, S) & \text{if } q \geq q_0
\end{cases}
\]

where \( \Delta(\varphi, r, s) \) is the objective function difference obtained by exchanging the facilities on locations \( r \) and \( s \) in solution \( \varphi \). \((R, S)\) is randomly chosen with a probability given by

\[
P_{(R,S)} = \frac{(\tau_{s0} + \tau_{s0})}{\sum_{u,v \in Q} (\tau_{u0} + \tau_{v0})} \quad \text{where } Q \text{ is the set of pairs } (u,v) \text{ satisfying } \Delta(\varphi, u,v) < 0.
\]

In effect, these pairs of facilities can improve the current solution \( \varphi \); \( q_0 \) decides how much randomness will be added into the algorithm. The smaller \( q_0 \) is, the more randomness is added; \( q \) is a random number chosen from \([0, 1]\). If \( q \geq q_0 \) then \((\tau_{s0} + \tau_{s0})\) is the heuristic for choosing the initial solution in the next iteration. \((\tau_{s0} + \tau_{s0})\) indicates the fitness of swapping the facilities on locations \( r \) and \( s \) according to the ants’ gained experience. The bigger \((\tau_{s0} + \tau_{s0})\) is, the more appropriate it is to do such a swap. So it is more probable to choose this solution to be the initial solution in the next iteration.

3.1.2 Forward-looking strategy in local search

In the QAP, the objective function difference \( \Delta(\varphi, r, s) \) obtained by exchanging facilities on locations \( r \) and \( s \) in solution can be computed in \( O(n) \). Specifically, the effect of a particular swap can be evaluated faster using information from preceding iterations. So, for a complete neighborhood scan, the first iteration in the local search is of complexity \( O(n^3) \) while the subsequent iterations can be done in \( O(n^2) \). To make full use of the lowered complexity of a neighborhood scan, a forward-looking strategy is proposed in this paper. Considering lines 2 to 5 in the 2-opt local search of algorithm 2, if there is no solution better than the current solution \( \varphi \) in its neighborhood, we do not stop the local search like the original one. Indeed, since there is probably a solution better than \( \varphi \) near its neighborhood, we make the algorithm search a step forward. That means choosing a detective solution \( \varphi' \) from the neighborhood of \( \varphi \), and then the best solution \( \varphi'' \) in the neighborhood of \( \varphi' \) is evaluated to see whether it is better than \( \varphi \). If \( \varphi'' \) is better than \( \varphi \), then the local search continues from \( \varphi'' \). Such a procedure is called one-time chance. Because of the formulation above, one-time chance only has a complexity of \( O(n^2) \). So with a little more time cost, this strategy enables the algorithm to leave local optima for better solutions with some probability. In this way, a local space will be searched more thoroughly and the algorithm’s searching ability will be strengthened. More precisely, the best solution in the neighborhood of \( \varphi \) is chosen as the detective solution \( \varphi' \). In fact, we only need to choose the pair \((r, s)\) which makes \( \Delta(\varphi, u, v) \) the smallest from \( \{(u, v)|1 \leq u < v \leq n\} \), and then exchange the facilities on this pair of locations to get the detective solution \( \varphi' \). The description of the improved 2-opt local search is presented in algorithm 4. Actually, in one-time chance if \( \varphi'' \) is not better than \( \varphi \), another solution can be chosen from the neighborhood of \( \varphi \) as the new detective solution \( \varphi' \), and again a procedure like one-time chance can be carried out. In this way, more-time chance can be implemented. However, as we noticed by preliminary experiments we conducted, although more-time chance can increase the probability to leave the local optima, it will require more time at each iteration of the search which will affect the overall time performance of the algorithm.

3.2. Improved tabu local search

Like in 2-opt local search, we have also added a random walk to the original tabu local search [7, 9] in order to avoid being trapped in a local optima. If there are neighboring solutions better than the current solution \( \varphi \), then choose the best neighboring solution for the next iteration with probability \( q_0 \), or choose the solution in the next iteration randomly from those better neighboring solutions with probability \( 1 - q_0 \). The detailed description of this strategy is as follows. When there are no neighboring solution better than the current solution \( \varphi \), then the best solution in the neighborhood will still be chosen in the next iteration like the original tabu local search. When there are solutions better than \( \varphi \) in its neighborhood, then choose a couple \((r, s)\) and get the solution in the next iteration by exchanging the facil-
Algorithm 4  1: Search the neighborhood of solution \( \varphi \), record the \( \Delta(\varphi, u, v), (1 \leq u < v \leq n) \).
2: if \( \exists \Delta(\varphi, u, v) < 0 \) then
3: Choose \((r, s)\) according to the formulation 1, and then exchange the facilities on locations \( r \) and \( s \) in the solution \( \varphi \).
4: Go back to Line 1.
5: else
6: Choose the best solution in the neighborhood of \( \varphi \) as the detective solution \( \varphi' \).
7: if the best solution \( \varphi'' \) in the neighborhood of \( \varphi' \) satisfies \( f(\varphi'') < f(\varphi) \) then
8: \( \varphi = \varphi'' \).
9: Go back to Line 1.
10: end if
11: end if
end

Algorithm 5  1: Search the neighborhood of solution \( \varphi \), record the \( \Delta(\varphi, u, v), (1 \leq u < v \leq n) \), and choose the best solution \( \varphi' \) among the non-tabu solutions and the solutions satisfying the aspiration criteria.
2: if \( \exists \Delta(\varphi, u, v) < 0 \) then
3: Choose \((r, s)\) according to the formulation 4, and then exchange the facilities on locations \( r \) and \( s \) in the solution \( \varphi \).
4: else
5: Choose \((r, s)\) which makes \( \Delta(\varphi, u, v), (1 \leq u < v \leq n) \) minimal, and then exchange the facilities on locations \( r \) and \( s \) in the solution \( \varphi \).
6: end if
7: Update the tabu lists.
8: if \( \text{Iteration}_{\text{max}} < \text{Iteration}_{\text{max}} \) then
9: Go back to Line 1.
10: end if
end

where \((R, S)\) is randomly chosen with a probability given by
\[
P_{(R, S)} = \log(1 - \Delta(\varphi, u, v)) / \sum_{(r, s) \in Q} \log(1 - \Delta(\varphi, u, v)).
\]
Here \( q_0 \), \( q \) and \( Q \) are defined in the same way as in the 2-opt local search. The description of the improved tabu local search is presented in algorithm 5. Here, \( \log(1 - \Delta(\varphi, u, v)) \) is used as the heuristic for choosing the solution in the next iteration. \( -\Delta(\varphi, u, v) \) means the cost decreases between the current solution and the solution obtained after the facilities swap. The larger \(-\Delta(\varphi, u, v)\) is, the smaller the cost of the new solution is (namely the better the new solution is). Thus it is more probable to choose this new solution for the next iteration of the local search. Since the relative difference \(-\Delta(\varphi, u, v)\) can be large, the function log is used to reduce such a difference to the log scale. Thus \( \log(1 - \Delta(\varphi, u, v)) > 0 \) makes sure that \( \log(1 - \Delta(\varphi, u, v)) > 0 \).

4. Experimentation

Problems from the well known QAPLIB are used here to evaluate the performance of the algorithms we have proposed and presented in this paper. We have chosen random problems from each of the following four classes of instances: unstructured, grid-based distance matrix, real-life instances and real-life-like instances. The detailed description about these classes can be found in [5].

The following parameter settings were used for each algorithm tested: \( m = 5 \), \( \rho = 0.8 \), \( \tau_{\text{min}} = 2/\text{est}_{\text{opt}}, \text{est}_{\text{opt}} \) is the objective function value of a randomly generated solution improved by local search, \( \tau_{\text{max}} = 5 * \tau_{\text{min}}, q_0 = 0.1 \). These parameters are set as optimal according to preliminary experiments we have conducted. For the same problem, the improved algorithm and original one were computed on the same Intel PC Pentium 4 computer running Windows XP system. All the procedures are coded in C. The experimental results, averaged over 20 runs, are reported in Tables 1 and 2. In Table 1, the improved MMAS\(_{2-opt}\) (that we call new MMAS\(_{2-opt}\)) has better performance (in terms of quality of the solutions returned, measured in percent above the best solution value known) on the four types of problems than the original MMAS\(_{2-opt}\). This improvement is even more significant in the case of the first two types of problems. This is justified by the fact that these problems are more difficult to solve i.e. the differences between the solutions obtained by the algorithms and the optimal solutions are so big that improvements can still be achieved. However, the third and fourth classes of problems are easy to solve so that both the improved MMAS\(_{2-opt}\) and the original MMAS\(_{2-opt}\) can find good solutions or optimal solutions in a short time. Similarly in Table 2, the improved MMAS\(_{\text{tabu}}\) (called new MMAS\(_{\text{tabu}}\)) also provides better results than the original MMAS\(_{\text{tabu}}\) for all four types of problems. We also noticed that the improvement in the case of the first two types of problems is more significant. When comparing the new MMAS\(_{2-opt}\) and the new MMAS\(_{\text{tabu}}\) we can see that MMAS\(_{\text{tabu}}\) has better performance in the first and second types of problems while MMAS\(_{2-opt}\) can find better solutions for the third and fourth types of problems except for Kra30b.

5. Conclusion

In this paper, two kinds of local search often employed in the Ant Colony Algorithm are improved in the case of the QAP problem. For the 2-opt local search, a random walk strategy is used to avoid a quick stagnation into local optima so that the solution space explored by local search will be extended. Additionally, a forward-looking strategy is proposed for 2-opt local search to leave some local optima with some probability, which makes the search more thoroughly.
Table 1. Quality of the MMAS\textsubscript{2-opt} and improved
MMAS\textsubscript{2-opt}.

<table>
<thead>
<tr>
<th>Instance n</th>
<th>Best known val</th>
<th>MMAS\textsubscript{2-opt}</th>
<th>New MMAS\textsubscript{2-opt}</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tai25a</td>
<td>25</td>
<td>1167256</td>
<td>1.660</td>
<td>1.381</td>
</tr>
<tr>
<td>Tai30a</td>
<td>30</td>
<td>1818146</td>
<td>1.532</td>
<td>1.228</td>
</tr>
<tr>
<td>Tai35a</td>
<td>35</td>
<td>2422002</td>
<td>1.835</td>
<td>1.633</td>
</tr>
<tr>
<td>Tai40a</td>
<td>40</td>
<td>3139370</td>
<td>2.100</td>
<td>1.733</td>
</tr>
<tr>
<td>Tai50a</td>
<td>50</td>
<td>4941410</td>
<td>2.422</td>
<td>2.027</td>
</tr>
</tbody>
</table>

Grid-based distance matrix

Nug25     | 25             | 3744                      | 0.095                         | 0.060   | 2       |
Nug30     | 30             | 6124                      | 0.548                         | 0.276   | 5       |
Sko42     | 42             | 15812                     | 1.249                         | 1.135   | 5       |
Sko49     | 49             | 25336                     | 1.211                         | 1.111   | 8       |
Sko56     | 56             | 34458                     | 1.339                         | 1.238   | 12      |

Real-life instances

Chr25a    | 25             | 3796                      | 4.742                         | 3.934   | 10      |
Kra30b    | 30             | 91420                     | 0.196                         | 0.125   | 10      |
Ste36b    | 36             | 15852                     | 0.187                         | 0.137   | 10      |
Bur26a    | 26             | 5426670                   | 0.034                         | 0.027   | 2       |

Real-life-like instances

Tai25b    | 25             | 344355646                 | 0.014                         | 0.009   | 5       |
Tai30b    | 30             | 637117113                 | 0.055                         | 0.033   | 10      |
Tai35b    | 35             | 283315445                 | 0.085                         | 0.063   | 15      |
Tai40b    | 40             | 637250948                 | 0.098                         | 0.075   | 20      |

Table 2. Quality of the MMAS\textsubscript{tabu} and improved MMAS\textsubscript{tabu}.

<table>
<thead>
<tr>
<th>Instance n</th>
<th>Best known val</th>
<th>MMAS\textsubscript{tabu}</th>
<th>New MMAS\textsubscript{tabu}</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tai25a</td>
<td>25</td>
<td>1472456</td>
<td>0.732</td>
<td>0.685</td>
</tr>
<tr>
<td>Tai30a</td>
<td>30</td>
<td>1818146</td>
<td>0.815</td>
<td>0.677</td>
</tr>
<tr>
<td>Tai35a</td>
<td>35</td>
<td>2422002</td>
<td>1.076</td>
<td>1.029</td>
</tr>
<tr>
<td>Tai40a</td>
<td>40</td>
<td>3139370</td>
<td>1.329</td>
<td>1.288</td>
</tr>
<tr>
<td>Tai50a</td>
<td>50</td>
<td>4941410</td>
<td>1.695</td>
<td>1.629</td>
</tr>
</tbody>
</table>

Total-based distance matrix

Nug25     | 25             | 3744                      | 0.034                         | 0       | 2       |
Nug30     | 30             | 6124                      | 0.189                         | 0.133   | 5       |
Sko42     | 42             | 15812                     | 0.548                         | 0.501   | 8       |
Sko49     | 49             | 25336                     | 0.548                         | 0.501   | 8       |
Sko56     | 56             | 34458                     | 0.633                         | 0.558   | 12      |

Real-life instances

Chr25a    | 25             | 3596                      | 4.490                         | 4.284   | 10      |
Kra30b    | 30             | 91420                     | 0.078                         | 0.065   | 10      |
Ste36b    | 36             | 15852                     | 1.275                         | 1.201   | 10      |
Bur26a    | 26             | 5426670                   | 0.056                         | 0.049   | 2       |

Real-life-like instances

Tai25b    | 25             | 344355646                 | 0.014                         | 0.018   | 5       |
Tai30b    | 30             | 637117113                 | 0.442                         | 0.392   | 10      |
Tai35b    | 35             | 283315445                 | 0.328                         | 0.208   | 15      |
Tai40b    | 40             | 637250948                 | 0.334                         | 0.388   | 20      |

For the tabu local search, a random walk strategy is also added into the algorithm. The experiments we conducted on problems taken from the QAPLIB library indicate that the improved local search techniques have obviously better performances than the original ones. This confirms the idea that a certain degree of random walk can help local search, in general, to find better solutions. Furthermore, because the local search is an independent step in the ant colony algorithm, the improvements we propose can be applied in ant colony algorithms other than the MMAS.

References


