A novel hybrid approach for solving the multiple traveling salesmen problem

Youssef Harrath, Abdul Fattah Salman, Abdulla Alqaddoumi, Hesham Hasan and Ahmed Radhi

Department of Computer Science, University of Bahrain, Zallaq, Bahrain

ABSTRACT

The multiple Travelling Salesmen Problem (mTSP) is one of the most popular and important operational research problems. It is a problem where \( n \) salesmen have to visit \( m \) cities such that each salesman has to visit at least one city and all the cities should be visited exactly once, starting and ending at one specific city. In this paper a new hybrid approach called AC2OptGA is proposed to solve the mTSP. AC2OptGA is a combination of three algorithms: Modified Ant Colony, 2-Opt, and Genetic Algorithm. Ant Colony-based algorithm is used to generate solutions on which the 2-Opt edge exchange algorithm is applied to enhance the obtained solutions. A Genetic Algorithm is then used to again improve the quality of the solutions. The reason behind combining the above-mentioned algorithms is to exploit their strengths in both global and local searches. The proposed approach is evaluated using various data instances from standard benchmarks. Using the TSPLIB benchmarks for large-sized instances, AC2OptGA shows better results than M-GELS, the current best known approach. For medium and small-sized data instances, AC2OptGA shows better results than other approaches and comparable results to M-GELS. Using the MTSP benchmarks (MTSP-51, MTSP-100 and MTSP-150), AC2OptGA outperforms other methods for number of salesmen less than 10 and is competitive with NMACO (BKS) for 10 salesmen.

1. Introduction

The Travelling Salesman Problem (TSP) is one of the most common and important optimization problems in operational research. In the literature, there are many variations of TSP that discuss different types of restrictions or constraints such as the number of salesmen, the number of starting/ending cities, time windows, etc. (Bektas, 2006). The multiple Travelling Salesman Problem (mTSP) is one of these variations, where disjoint routes for \( m \) salesmen travelling to \( n \) cities have to be found. The routes should cover all cities and every salesman should visit at least one city, originating from and returning to the same unified start city. The quality of the routes can be evaluated using qualitative and quantitative measures, such as distance, time, cost, customer satisfaction, salesman qualification, workload balance to name a few. In the literature, many studies have been conducted to solve the TSP with its variations, and relatively few approaches were applied to handle the mTSP.

The importance of the mTSP arises from the fact that it is a well-known combinatorial optimization problem that models many real-life problems such as the coordination of global navigation satellite systems, crew and maintenance scheduling, mission planning, interview scheduling, vehicle routing, etc.

The mTSP has been tackled using many different approaches; most of them are nature-inspired algorithms such as the Ant Colony Optimization (ACO), Genetic Algorithms (GA) and Gravitational Emulation Local Search (GELS). In addition, nearest-neighbour based search algorithms such as 2-Opt, were frequently used to improve the quality of solutions obtained by other algorithms. These methods proved to be efficient and provide reasonable solutions when implemented on their own. To the best of our knowledge, no approach combining the above mentioned algorithms has been applied to solve the mTSP. To exploit the benefits of ACO, 2-Opt, and GA algorithms, the authors propose to combine them into a new hybrid approach, AC2OptGA.

The basic idea of solving the mTSP using Ant Colony Optimization is that the salesmen in their tours imitate ants in their search for food. Ants go separately and deposit a pheromone in the trail they follow when searching for food. The more the
pheromone in a given trail, the more likely that other ants will follow it. Pheromone evaporates over time, so a pheromone in a longer path to the food source will evaporate before a pheromone in a shorter path. Ants will eventually follow the path with the stronger pheromone, therefore the shorter the path. ACO algorithms use artificial ants to find the shortest path (or tours in mTSP) in a graph. A basic ACO algorithm was applied to an mTSP in Krishna, Ravindraandra, and Gupta (2015). Majid, Farzad, and Farhad (2013) proposed a new modified ACO algorithm that disposes more pheromone for the best solution to prevent divergence to local minimum.

2-Opt is a simple local search algorithm used to improve solutions of TSP problems. The essence of 2-Opt is to select two edges from a route in a graph based on certain criteria. 2-Opt replaces the selected edges with their corresponding crossing edges such that the objective value of the new route is enhanced.

Genetic Algorithms are powerful nature-inspired optimization techniques based on randomized processes such as selection and reproduction. GA are used to solve many complex problems such as TSP in reasonable time. GA generally have good performance due to their parallel search features where many solutions are combined iteratively until a suitable solution is generated.

The aim of this paper is to propose a new hybrid algorithm combining the aforementioned three algorithms to solve the mTSP with the objective of minimizing the total distance travelled by all salesmen. The rest of the paper is organized as follows: In section 2 we review the main approaches used to solve the single-objective mTSP. The proposed approach to solve mTSP, AC2OptGA, is presented in section 3. Next, in section 4, we discuss the results obtained from benchmarking AC2OptGA using different-sized data instances and compare them with the results of the best known approaches in the literature. Section 5 summarizes the main conclusions achieved by this work.

2. Literature review

The mTSP is classified as NP-Hard and the performance of algorithms that produce exact solutions depends on the number of visited cities and usually requires an exponential amount of time with an increase in the number of visited cities. Exact and heuristic-based approaches have been used to solve the mTSP. The exact approaches guarantee to find the optimal solution for small-sized instances only, but when the problem size is large it would take exponential time to find an optimal solution. Heuristic-based approaches provide near-optimal solutions for large-sized instances in a reasonable time.

2.1. Exact approaches

The exact approaches are based on either transforming the mTSP to TSP, or on the relaxation of some constraints of the problem Bellmore and Hong (1974) and Bektas (2006).

The first attempt to solve the mTSP without transformation to standard TSP is suggested by Laporte and Nobert (1980), who proposed two exact algorithms: a straight algorithm and a reverse algorithm. The straight algorithm is based on initial relaxation of Subtour Elimination Constraints (SEC). If a violation is found, a constraint associated with each subtour is introduced into the problem to remove the violating subtour. The reverse algorithm also relaxes SEC, but it differs by checking the integrity before a solution is found.

Ali and Kennington (1986) proposed a branch-and-bound algorithm which uses Lagrangian relaxation and a subgradient algorithm to solve Lagrangian dual, a problem which is modelled as an asymmetric mTSP. The algorithm can be applied to both symmetric and asymmetric mTSPs. Gromicho, Paix, and Bronco (1992) proposed another exact algorithm for asymmetric mTSPs. It uses quasi-assignment (QA) relaxation of certain SEC in the problem. They also embedded additive bounding procedures to improve the lower bound results, for which QA relaxation does poorly.

Gavish and Srikanth (1986) developed the first algorithm to solve large-scale mTSP. They used a branch-and-bound based method while establishing lower bounds through Lagrangian relaxation. The
authors concluded that their algorithm performs better than Laporte and Nobert (1980) and Ali and Kennington (1986).

### 2.2. Heuristic-based approaches

In this section, we briefly describe some of heuristic-based approaches. Bektas (2006) surveys the transformation-based approaches to solve the mTSP.

Lixin, Jiyin, Aiying, and Zihou (2000) proposed a Modified Genetic Algorithm (MGA) for hot-rolling scheduling in Shanghai Baoshan Iron and Steel Complex. The researchers concluded that MGA was constructed to obtain near-optimal solutions and showed 20% improvement over the manual system.

Carter and Ragsdale (2006) used a genetic algorithm-based technique to solve the mTSP. The authors conducted a study of the performance of the existing chromosome representations and proposed a new two-part chromosome representation. They concluded that its search space is smaller than the previous chromosome representations. The researchers also found that the two-part chromosome representation is superior as the number of salesmen increased especially when individual salesmen tours have to be shortened in order to balance their tour length.

Kiraly & Abonyi (2011) proposed a multi-chromosome genetic algorithm by allocating one chromosome for each salesman. New mutation operators such as in-route mutations and the cross-route mutations were implemented. The authors found that the multi-chromosome technique converges to the optima faster than the two-part technique and reduces the overall search space for the problem. The authors claimed that this approach is more effective than any other GA approach to date.

Blum (2005) explained how the ACO can be applied to continuous optimization and discussed the new trend in ACO that uses a hybrid optimization technique to solve the mTSP. Junjie and Dingwei (2006) attempted to solve a constrained version of mTSP that has an upper bound on the maximum number of cities a salesman can visit. The solution to the mTSP is developed using an ACO algorithm and its performance is tested using benchmarks from TSPLIB. The researchers found that the modified Genetic Algorithm (MGA) performed slightly better than their algorithm for small data sets, but their algorithm outperformed the MGA algorithm for large data sets. A new ACO algorithm NMACO to solve mTSP was presented in Majid et al. (2013). NMACO improved the performance of the classic ACO by escaping from local optimum points and hence generated better solutions. Also, compared with Sweep Algorithm + Elite Ant System algorithm and Modified Ant Colony Algorithm, NMACO produced better results in five of the six test cases. Krishna et al. (2015) proposed another ACO approach that can be applied to solve the mTSP with ability constraints. The authors avoided stagnation and premature convergence by using distribution strategy of initial ants and dynamic heuristic parameter updating based on entropy. The experimental results and the performance comparison showed that the proposed approach produced better results than the ACO algorithm.

Aditi, Manas, and Manoranjan (2016) formulated some TSPs as linear programmes with imprecise data. The cost and time, or both are minimized by a hybrid heuristic algorithm combining the ACO and GA. The authors developed an algorithm that is capable of solving single- and multi-objective constrained large TSPs with crisp, fuzzy and rough data. The researchers claim that their algorithm outperforms other conventional non-hybrid algorithms and can be used to accommodate many different real-life scenarios. Sedighpour, Yousefkhoshbakht, and Mahmoodi (2012) proposed a hybrid meta-heuristic algorithm GA2OPT consisting of the modified genetic algorithm and the 2-opt local search. In each iteration, the authors first applied a modified genetic algorithm to obtain a solution and then used the 2-opt local search to improve the results and increase the performance of the algorithm. GA2Opt was compared with different meta-heuristic algorithms such as the MGA and the Modified Ant Colony Algorithm (MACA). The results showed that GA2OPT had the ability to find better solutions than MGA and MACA in some of the instances and worse in other instances. Sho, Haruna, and Yoshifumi (2011, September) proposed a hybrid algorithm GIACO consisting of genetic algorithm and Intelligent and Dull Ant Colony Optimization (IDACO). The intelligent ants trail the pheromone and use genetic information, whereas the dull ants, a result of the mutation of the GA, cannot trail the pheromone nor use the genetic information. The authors claim that IDACO with the dull ants produced better results than the conventional ACO, which consists of the intelligent ants only. GIACO was compared with ACO, GA and the GA-ACO, which consists of the intelligent ants only. The authors claimed that GIACO obtained better results than GA-ACO, which itself obtained better results than the traditional ACO and GA algorithms. Shokouhi, Farahnaz, Hengameh, and Hosseinabadi (2015) proposed a modified Gravitational Emulation Local Search (M-GELS) algorithm to solve the symmetric mTSP. Using this approach, the sweep algorithm is used to generate a set of feasible solutions that will be improved by the M-GELS. The authors concluded that M-GELS has superiority over
the well-known optimization algorithms for solving the mTSP. Yousefikhoshbakht and Sedighpour (2012) proposed SW + AEelite, a combination of the sweep algorithm, the elite ant colony algorithm and 3-opt local search algorithms. The authors used the sweep algorithm to generate a set of feasible solutions later improved by the elite ant system and the 3-opt local search. They stated that the strength of this algorithm is that it takes a short time and few iterations to gain a good solution. SW + AEelite was compared with MGA and MACO algorithms and yields better solutions in large-scale problems compared with MGA. Compared with MACO, SW + AEelite was superior in all instances except pr226.

3. Methodology

Many approaches have been used to tackle the mTSP problem. Most of these approaches did not solve optimally the problem. We propose a new hybrid approach based on 2-opt local search and the heuristic-based methods ACO and GA. The approach iteratively applies ACO, 2-opt and GA in order to find the best possible feasible solution. ACO is used to generate a population of good feasible solutions for the TSP (one salesman). 2-Opt is then applied to improve these solutions. After that, every TSP solution is randomly broken down into an mTSP solution. Then, one step of GA is applied to the previously obtained solutions. In this step, suitable selection, crossover and mutation operators are applied to improve the mTSP solutions. The Pheromone map is then updated to guide the next iteration.

Algorithm 1 presents the detailed steps of the pseudo-code of the new hybrid approach AC2OptGA.

3.1. Controlling parameters

There are a number of control parameters that are used to tweak the output of the proposed approach including:

- \( s \) : Population size (number of ants in ACO or solutions in GA)
- \( \alpha \) : Pheromone exponential weight
- \( \beta \) : Heuristic exponential weight
- \( Q \) : Pheromone strength
- \( \rho \) : Evaporation rate
- \( T_{\text{cond}} \) : Number of iterations

\( \alpha, \beta \) are the parameters set by user \textit{a priori} to control the weight/influence of the amount of pheromone and the inverse of cost to visit node \( a \) hold in the formula, respectively.

3.2. Ant solution construction

The proposed approach begins by constructing initial TSP solutions to create a population \( P \) using the ACO construction method. An ant here is an agent

Algorithm 1 Load Balance Algorithm (LBA): Assignment of cities to the salesmen.

1: procedure AC2OptGA:
2: SET Controlling parameters;
3: repeat
4: CONSTRUCT population \( P \) having \( s \) TSP solutions using ACO
5: APPLY 2-opt Edge Exchange to all solutions in \( P \)
6: BREAK DOWN each TSP solution \( S_i \in P \) into \( m \) mTSP tours;
7: EVALUATE the fitness of all solutions \( S_i \in P \)
8: \( S_b \leftarrow \) best solution in \( P \)
9: Group the solutions \( S_i \in P \) in Groups of 8
10: for every group of eight solutions do
11: FIND the best solution
12: APPLY eight GA crossover and mutation operators to the best solution to generate 8 new solutions
13: INSERT the eight new solutions in a new population \( P' \)
14: end for
15: \( P \leftarrow P' \)
16: EVALUATE the fitness of all members of \( P \)
17: \( S_b \leftarrow \) best solution in \( P \)
18: EVAPORATE pheromone on all edges
19: if cost of \( S_b \) < cost of \( S_b \) then
20: DEPOSIT pheromone on \( S_b \) edges
21: obs \leftarrow S_b
22: else
23: DEPOSIT pheromone on \( S_b \) edges
24: obs \leftarrow S_b
25: end if
26: if cost of obs < cost of \( b_{m+1} \) then
27: \( b_{m+1} \leftarrow \) obs
28: end if
29: DEPOSIT pheromone on \( b_{m+1} \) edges
30: until \( T_{\text{cond}} \) is satisfied
31: end procedure
that determines a single feasible route which traverses all the cities based on a heuristic probability function as shown in the next equation. City selection is an iterative process that uses the same function at every step in order to determine the city to be visited next. The function outputs a scalar probability value that the \( k^{th} \) ant uses to determine how “likely” it is for it to move from node \( i \) to node \( j \) that has not been visited yet by that ant. It is defined as follows:

\[
p^k_{ij} = \begin{cases} 
(\tau^k_{ij} \eta^k_{ij}) & \text{if } j \in N^k_i \\
0 & \text{if } j \not\in N^k_i 
\end{cases}
\]

where:

- \( N^k_i \): The set of nodes adjacent to node \( i \) that have not been visited yet by the \( k \)th ant;
- \( \tau^k_{ij} \): The amount of pheromone on edge connecting node \( i \) to \( j \);
- \( \eta^k_{ij} \): Known as “Heuristic Information”, is the inverse of cost to visit node \( j \) from node \( i \). If this was used alone, the ants exhibit greedy behaviour similar to Nearest Neighbour algorithm;
- \( \alpha, \beta \): Parameters set by user \textit{a priori} that control the weight/influence \( \tau^k_{ij} \) and \( \eta^k_{ij} \) hold in the formula, respectively.

We conclude that the heuristic probability function shown in the above equation selects the next node to be visited based on two important factors: 1) the amount of pheromone on edge connecting the two cities, and 2) the cost of traversing an edge to the next city. The function will be calculated and applied \( n-1 \) times by each ant to generate a complete route. A total of \( s \) ants will leave the depot node and construct a complete route using the probability function \( p^k_{ij} \). Note that the depot node is always fixed for all ants. Therefore, \( s \) solutions will be generated and populate the solution set \( P \).

### 3.3. 2-opt edge exchange

The population \( P \) contains a set of all solutions constructed using ACO from the previous step. ACO solution construction is not guaranteed to pick the most optimized solutions in its neighbourhood since the routes generated are obtained by the probability function \( p^k_{ij} \). The 2-opt Edge Exchange algorithm is then applied on the solutions in \( P \) for further enhancements.

2-opt Edge Exchange does a pairwise exchange of edges in a solution with the goal of attaining a new route ordering which has a lower cost than the original input route. The proposed algorithm applies 2-opt to all edges (except those associated with depot node). When 2-opt has been applied to all edges in the route, the resulting route is called 2-optimal. The pseudo-code for the 2-opt Edge Exchange algorithm implemented in AC2OptGA is as follows:

Figure 1 illustrates a sample of exchanging the edges \((ab, cd)\) with the edges \((ad, bc)\) leading to an improvement of two units.

### 3.4. TSP to mTSP route breaking

In this stage, the proposed algorithm AC2OptGA transforms every TSP solution in population \( P \) into an mTSP solution. This is done by randomly choosing \( m-1 \) cutpoints in the TSP solution \( S_i \in P \) and then assigning each part of the TSP solution to a salesman. We ensure in this step that every salesman has a valid tour with respect to the number of nodes to be visited.

The proposed algorithm uses the multi-chromosome representation technique suggested by Kiraly & Abonyi (2011), who empirically prove that this representation of the mTSP produces the smallest complete search space by eliminating redundancies and simplifies the chromosome initialization with node values. The size of the search space of multi-chromosome representation technique is \( n!((m-1)) \). Each chromosome represents a salesman tour as an ordered list of nodes to be visited. The lists do not contain starting city node as the depot is fixed for all salesmen.

![Algorithm 2 Pseudocode of 2-Opt Edge Exchange](image-url)
3.5. Genetic algorithm

The Genetic Algorithm is applied to each group of eight mTSP solutions in \( P \). The fitness function calculates the cost of each mTSP solution as the summation of Euclidean distances between nodes involved in the solution. The GA will select the least expensive solution (parent) among the group and apply eight different crossover and mutation operators on the parent to produce eight new solutions (children) that will be inserted in the new population \( P' \). New forms of crossover and mutation operators have been introduced by Kiraly & Abonyi (2011). The new mutation operators include Flip, Swap, and Slide. The new crossover operators include one-point crossover, flip and crossover, swap and crossover and slide and crossover.

3.6. Pheromone update

Ants in real life deposit pheromones on their way back from the food source to inform other ants about the route to follow. The amount of pheromone deposited is the inverse of the total route length, i.e., the shorter the path to the food source, the higher the amount of deposited pheromone, hence being a more “attractive” region for the ants. A longer path will deposit a lower amount of pheromone, thus the ants will find it less “attractive”. Over time, the pheromone evaporates to exclude longer and unused paths.

The pheromone evaporates over time and its amount left on each edge is calculated by a mathematical formula that is applied to all edges of every solution \( S_t \in P \). With pheromone evaporation, the algorithm converges towards shorter and more “fresh” solution edges and tends to “forget” longer edges of older solutions. This allows the ants in the solution construction phase “to favour exploration of new areas in the solution space” (Blum, 2005).

The amount of pheromone on an edge connecting nodes \( i \) and \( j \) after evaporation is calculated as follows:

\[
\tau_{ij} = (1 - \rho) \cdot \tau_{ij}
\]

where \( \rho \in (0, 1] \) is a parameter that controls the evaporation rate of pheromone. The higher the evaporation rate, the more pheromone is removed in each iteration and vice versa.

The amount of pheromone deposited on an edge connecting nodes \( i \) and \( j \) is calculated as follows:

\[
\Delta \tau_{ij}^t = \begin{cases} 
Q / L_t & \text{if edge } ij \text{ is used} \\
0 & \text{otherwise}
\end{cases}
\]

where:

- \( t \): Identifies solution whose pheromone deposit value is to be determined. In the proposed algorithm, \( t \) is either the current iteration best solution or the global best solution;
- \( \tau_{ij}^t \): Pheromone on an edge \( ij \);
- \( L_t \): Total route length \( t \);
- \( Q \): A positive constant parameter that controls the strength of pheromone deposits. The higher the value of \( Q \), the larger is the resulting pheromone deposit.

Pheromones are deposited on all the edges of the best known solution, \( bks \), and current best solution, \( cbs \), only. This is a variation of the Elitist Ant System introduced in Dorigo (1992) and (suggested/improved) by Majid et al. (2013). The pheromone deposits on \( bks \) edges seek to lead the algorithm to discover better solutions. Pheromone deposits on \( cbs \) edges prevent premature convergence toward local minima that may improve \( bks \) by providing suggestions located in \( cbs \) neighbourhood. Overall, the pheromone update process is done in an elitist style, favouring only the best solutions.

The pheromone deposit formula for an edge \( ij \) implemented in the proposed algorithm is given as follows:
The proposed algorithm AC2OptGA is implemented and tested using MATLAB R2012a on 14 PCs having the following specifications: 3.33GHz Intel® Core™ i5-660 CPU, 4.00 GB RAM and running under Windows 7. AC2OptGA is evaluated using literature-standard benchmarks taken from the library TSPLIB used for testing the performance of algorithms developed to solve TSP and mTSP. The algorithm is run on small-sized problems (Pr76 and Pr152), medium-sized problems (Pr226 and Pr299), and large-sized problems (Pr439 and Pr1002). In order to ensure that the workload is fairly distributed among the salesmen, a parameter $u$ is introduced. This parameter designates the maximum number of cities to be visited by each salesman.

The proposed algorithm is run with a fixed number of salesmen ($m = 5$) for all instances. The following values of the controlling parameters are obtained empirically: population size $a = 3$, pheromone strength $\alpha = 100$, heuristic exponential weight $\beta = 3$, pheromone strength $\varphi = 0.29$, and number of iterations $t = 2000$.

The AC2OptGA is tested by making 10 independent runs in every instance. The distances of the best solution (BS) and the average of solutions (AS) for each problem size are summarized in Table 1. To the best of our knowledge, the results produced by AC2OptGA are compared to those obtained by the algorithms producing the best results according to the literature. These algorithms are: Modified Sweep and Ant Colony Algorithm (SW + $A_{elite}$) (Yousefikhoshbakht & Sedighpour, 2012), Modified Genetic Algorithm (MGA) (Lixin et al., 2000), Novel Modified Ant Colony Optimization (NMACO) (Majid et al., 2013), and Modified Gravitational Emulation Local Search (M-GELS) (Shokouhi et al., 2015). There is no available information (NA) for the distance of the best solutions for M-GELS.

Table 2 shows the percentage enhancement of AC2OptGA compared to the other algorithms. A

| Instance | $n$ | $u$ | BS | AS | BS | AS | BS | AS | BS | AS | BS | AS |
|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Pr76     | 76  | 20  | 159289 | 163120 | 157495 | 157562 | NA | 147734 | 157444 | 160574 | 157413 | 157499 |
| Pr152    | 152 | 40  | 127520 | 132057 | 127791 | 128004 | NA | 119205 | 127839 | 133337 | 127781 | 127988 |
| Pr226    | 226 | 50  | 161084 | 163235 | 167665 | 168156 | NA | 160350 | 166827 | 178501 | 167239 | 167821 |
| Pr299    | 299 | 70  | 77810  | 80247  | 81998  | 82195  | NA | 76654  | 82176  | 85796  | 81261  | 82012 |
| Pr439    | 439 | 100 | 149675 | 153199 | 161725 | 162657 | NA | 155523 | 173839 | 183698 | 160298 | 161687 |
| Pr1002   | 1002| 220 | 351371 | 355790 | 379871 | 381654 | NA | 356341 | 427269 | 459179 | 379042 | 381108 |

3.7. Termination condition

The mTSP is classified as an NP-hard problem, hence finding an optimal solution especially for big data instances requires an exponential amount of time. There are many criteria to terminate algorithms used to solve such classes of problems. The following are examples of termination criteria: reaching a pre-set goal of total distance, elapsing a maximum amount of time, reaching a pre-defined number of iterations, or reaching a point at which no improvements to the solution are obtained. Finding a suitable stopping criterion becomes important to ensure that the algorithm produces a near-optimal solution and runs for a reasonable amount of time.

AC2OptGA runs for a certain amount of iterations, determined empirically, then is stopped and the global best solution obtained, $bks$, is considered to be the final solution.

4. Implementation and results

The proposed algorithm AC2OptGA is implemented and tested using MATLAB R2012a on 14 PCs having the following specifications: 3.33GHz Intel® Core™ i5-660 CPU, 4.00 GB RAM and running under Windows 7.
negative value in the table indicates a better performance of AC2OptGA compared with the other algorithms.

According to Tables 1 and 2, the proposed algorithm AC2OptGA produces better results than all the above mentioned algorithms for medium- and large-sized data instances, except M-GELS that produces better results for small- and medium-sized data instances. AC2OptGA is competitive with all mentioned algorithms for small-sized data instances.

Table 3 presents the percentage difference between the average and best solutions produced

Figure 3. Best, average and worst performance of AC2OptGA for small instances.

Figure 4. Best, average and worst performance of AC2OptGA for medium-sized instances.

Figure 5. Best, average and worst performance of AC2OptGA for large-sized instances.
Table 4. Performance Comparison of AC2OptGA, GA, GGA, NGGA, ACO, and NMACO (BKS).

| Instance | m | n | GA | GGA | NGGA | ACO | NMACO | AC2OptGA | % Diff. with BKS |
|----------|---|---|----|-----|------|-----|-------|----------|------------------|
| MTSP-51  | 3 | 51 | 460| 924 | 543  | 449 | 447   | 424      | −5.57%           |
| 10       | 669| 1000| 723| 592  | 586  | 481 | 473   | 460      | −4.37%           |
| MTSP-100 | 3 | 22959| 79347| 22241| 22051| 21472|       |          | −2.63%           |
| 5        | 24559| 70871| —   | 23901| 23678| 23073| —     |          | −2.56%           |
| 10       | 33136| 89778| —   | 27981| 27643| 28197| —     |          | 0.00%            |
| 20       | 62963| 137805| —   | 39991| 39731| 46526| —     |          | 17.10%           |

Table 5. Performance Comparison between AC2Opt and AC2OptGA.

| Instance | m | n | AC2Opt | AC2OptGA | Percentage Difference |
|----------|---|---|--------|----------|-----------------------|
| Pr76     | 5 | 163650| 159289| 4.2%     |
| Pr152    | 129670| 127520| 1.7%     |
| Pr226    | 159685| 161084| 0.9%     |
| Pr299    | 78216| 78710| 0.6%     |
| Pr439    | 149293| 149675| 0.3%     |
| Pr1002   | 356478| 351371| 1.4%     |

Table 6. Execution times in seconds for ACO, MGA, AC2Opt, and AC2OptGA.

| Instance | m | n | ACO | MGA | AC2Opt | AC2OptGA |
|----------|---|---|-----|-----|--------|----------|
| Pr76     | 5 | 43 | 46  | 57  |
| Pr152    | 128| 91 | 146 | 156 |
| Pr226    | 143| 165| 313 | 319 |
| Pr299    | 288| 363| 589 | 599 |
| Pr439    | 563| 623| 1486| 1758|
| Pr1002   | 2620| 2892| 3934| 6706|

NMACO Majid et al. (2013) as shown in Table 4. From that table, AC2OptGA outperforms GA, GGA and NGGA for all tested instances and different number of salesmen. Moreover, the proposed algorithm produces better results than ACO and NMACO for all instances with $m = 3$ and $m = 5$, and MTSP-51 with $m = 10$. In addition to that, AC2OptGA is competitive with ACO and NMACO for the instances MTSP-100 and MTSP-150 for $m = 10$ with a percentage difference of 2% and 3.99%, respectively. For $m = 20$ and $m = 30$, ACO and NMACO produce better results.

Experiments were conducted to investigate the effect of adding GA to AC2Opt. AC2Opt is obtained by removing the GA part (steps 9–18 in Algorithm 1) from AC2OptGA. Table 5 summarizes the results obtained from running both algorithms on 18 benchmarks of different sizes and different numbers of salesmen. AC2OptGA demonstrates better performance for 16 instances and shows up to 8% improvement over AC2Opt. AC2Opt algorithm produced slightly better performance of less than 1% for only two instances.

When evaluating the performance of algorithms for solving static problems like the mTSP problem in hand, the running time is mostly not considered as an important factor. Once the final solution is obtained and adopted, the algorithm will not be run as long as the problem parameters remain unchangeable. Moreover, the mTSP belongs to a very complex class of combinatorial problems and the exact methods such as dynamic programming and branch and bound take too much time to find an optimal solution for even small instances, whereas the heuristic methods like ACO, GA, and 2-Opt provide very good solutions in an accepted running time for even very large instances. Running time can be considerably reduced by using super parallel computers.

Table 6 summarizes the execution times for ACO, MGA, AC2Opt, and AC2OptGA. By AC2OptGA, SW + ASElite, MGA, and NMACO. Best solutions for M-GELS are not available. As shown in this table, the values are very small for each algorithm, which means that the proposed AC2OptGA algorithm is competitive with the indicated algorithms. In addition, for all runs, AC2OptGA algorithm produces consistent solutions close to the best solution.

Figures 3–5 illustrate that the differences between the worst, average, and best solutions for the proposed AC2OptGA algorithm are very close and do not exceed 6% for all benchmarks, which again ensures the consistency of the solutions obtained by AC2OptGA.

Other mTSP instances (MTSP-51, MTSP-100 and MTSP-150) are used to test the performance of AC2OptGA with different numbers of salesmen. The obtained results are compared with five other methods: GA Singh, Singh, and Prasad (2018), GGA, NGGA, ACO and the Best Known Solution (BKS) by AC2OptGA, SW + ASElite, MGA, and NMACO. Best solutions for M-GELS are not available.
ACOptGA encourages exploration of good solutions while avoiding convergence towards local minima. 2-opt improves the overall solution quality, which is in turn sent to GA operators for even further enhancements that are absorbed in every iteration by the ants.

5. Conclusion

In this paper, a new approach, AC2OptGA, for solving the Multiple Travelling Salesmen Problem is presented and implemented using Matlab. The proposed AC2OptGA approach combines the modified Ant Colony Optimization, the 2-Opt local search heuristic, and the Genetic Algorithm. An experimental study was conducted to evaluate the performance of the proposed approach to solve various mTSP instances of different sizes and different salesmen. The proposed approach was compared with the main approaches published that solve the mTSP. The obtained results show that the AC2OptGA provided better solutions for mTSPs with large- and medium-sized data instances and competitive results for small-sized data instances for the Pr76-Pr1002 from the TSPLIB. For the MTSP-51, MTSP-100 and MTSP-150 benchmarks, AC2OptGA outperforms other methods for number of salesmen less than 10.

Additionally, the new approach produces a wide range of high-performance solutions with small differences between the best, average, and worst. We believe that there is still room for improvement by incorporating smart techniques to build better mTSP solutions by applying the gravitational emulation techniques.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Aditi, K., Manas, M. K., & Manoranjan, M. (2016). A hybrid heuristic algorithm for single and multi-objective imprecise traveling salesman problems. Journal of Intelligent and Fuzzy Systems, 30(4), 1987–2001.

Ali, A. I., & Kennington, J. (1996). The asymmetric M-traveling salesmen problem: A duality based branch-and-bound algorithm. Discrete Applied Mathematics, 13 (2), 259–27. doi:10.1016/0166-218X(86)90087-9

Bektas, T. (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. Omega, 34(3), 209–219. doi:10.1016/j.omega.2004.10.004

Bellmore, M., & Hong, S. (1974). Transformation of multisalessman problem to the standard traveling salesman problem. Journal of the ACM, 21(3), 500–504. doi: 10.1145/321832.321847

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. Physics of Life Reviews, 2(4), 353–373. doi:10.1016/j.plrev.2005.10.001

Carter, A. E., & Ragsdale, C. T. (2006). A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European Journal of Operational Research, 175(1), 246–257. doi:10.1016/j.ejor.2005.04.027

Dorigo, M. (1992). Optimization, Learning and Natural Algorithms (Ph.D. Thesis). Politecnico di Milano, Italy.

Gavish, B., & Srikanth, K. (1986). An optimal solution method for large-scale multiple traveling salesmen problems. Operations Research, 34(5), 698–717. doi: 10.1287/opre.34.5.698

Gromicho, J., Paix, J., & Bronco, I. (1992). Exact solution of multiple traveling salesman problems. Combinatorial optimization: New Frontiers in theory and practice (pp. 291–292). Heidelberg, Germany: Springer Berlin Heidelberg.

Junjie, P., & Dingwei, W. (2006). An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem. Proceedings of the First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC’06) (pp. 210–213). Beijing, China.

Király, A., & Abonyi, J. (2011). Optimization of multiple traveling salesmen problem by a novel representation based genetic algorithm. In Intelligent computational optimization in engineering (pp. 241–269). Berlin, Germany: Springer.

Krishna, H., Ravindraand, K., & Gupta, S. (2015). An approach for solving multiple traveling salesman problem using ant colony optimization. Computer Engineering and Intelligent Systems, 6(2), 13–17.

Laporte, G., & Nobert, Y. (1980). A cutting planes algorithm for the M-salesmen problem. Journal of the Operational Research Society, 31(11), 1017–1023. doi:10.2307/258128

Lixin, T., Jiying, L., Aijing, R., & ong., & Zihou, Y. (2000). A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. European Journal of Operational Research, 124(2), 267–282.

Majid, Y. K., Farzad, D., & Farhad, R. (2013). Modification of the ant colony optimization for solving the multiple traveling salesman problem. Romanian Journal of Information Science and Technology, 16(1), 65–80.

Sedighpour, M., Yousefikhoshbakht, M., & Mahmoodi, D. N. (2012). An effective genetic algorithm for solving the multiple traveling salesman problem. Journal of Optimization in Industrial Engineering, 4(8), 73–79.

Sho, S., Haruna, M., & Yoshifumi, N. (2011). Ant Colony Optimization using Genetic Information for TSP. Proceedings of the International Symposium on Nonlinear Theory and its Applications OALTA2011 (pp. 48–51). Kobe, Japan.

Shokouhi, R. A., Farahnaz, M., Hengameh, K., & Hosseinibadi, A. R. (2015). Solving multiple traveling salesman problem using the gravitational emulation local search algorithm. Appl. Math, 9(2), 699–709.

Singh, D. R., Singh, M. K., Singh, T., & Prasad, R. (2018). Genetic algorithm for solving multiple traveling salesmen problem using a new crossover and population generation. Computación y Sistemas, 22(2), 491–503.

Yousefikhoshbakht, M., & Sedighpour, M. (2012). A combination of sweep algorithm and elite ant colony optimization for solving the multiple traveling salesman problem. Proceedings of the Romanian Academy - Series A, 13(4), 295–301.