The Nearest Neighbor Ant Colony System: A Spatially-Explicit Algorithm for the Traveling Salesman Problem

Jean-Claude Thill and Yu-Cheng Kuo

Abstract Inspired from the behavior of real ants, the ant colony algorithm has provided a new approach for solving discrete optimization problems, such as the traveling salesman problem. However, traditional ant colony algorithms may consume an inordinate amount of computing time to converge to a solution. In this chapter, a new heuristic algorithm called the nearest neighbor ant colony system (NNAC) is proposed in order to reduce computing time, without sacrificing on the optimality properties of the solutions. The NNAC is an intelligent form of the original ant colony system that follows a spatial strategy. Thanks to a search strategy that eliminates a large number of inefficient solutions up front on the basis of proximity-based neighborhoods, the NNAC is able to find the best solution in a fraction of the time that the conventional ant colony system consumes. The paper summarizes the principles of heuristics based on ant colony systems and highlighted some of their limitations. The proposed NNAC algorithm is presented in detail. The NNAC is tested on five different data sets and compared to a traditional ant colony system heuristic.

Keywords TSP • Traveling Salesman Problem • Ant colony system • Spatially explicit modeling

Introduction

The Traveling salesman problem (TSP) is one of the most notable operational models of spatial planning. It involves finding the shortest way of visiting each of a given set of locations (also known as cities) exactly once and returning to the

J.-C. Thill (✉)
Department of Geography and Earth Sciences, University of North Carolina, Charlotte, NC 28223, USA
e-mail: jean-claude.Thill@uncc.edu

Y.-C. Kuo
Department of Geography, University at Buffalo – The State University of New York, Buffalo, NY, USA

J.-C. Thill (ed.), *Spatial Analysis and Location Modeling in Urban and Regional Systems*, Advances in Geographic Information Science,
https://doi.org/10.1007/978-3-642-37896-6_13

starting point at the end of the tour. If travel cost is substituted for the distance between each pair of locations, the TSP is then to find the least-cost way of visiting all the locations.

The TSP has a number of real-world applications such as bus routing (Spada et al. 2005), delivery and repair vehicle routing (Weigel and Cao 1999), newspaper delivery (Song et al. 2002), business logistics (Exnar and Machac 2011), and individual trip planning (Thill and Thomas 1991). Because it is such a prominent problem of spatial planning and since many problems can be structured as a TSP on networks, the TSP is fairly widely implemented in Geographic Information Systems (Curtin et al. 2014). There are also a number of applications beyond these conventional applications. In industrial engineering and in sciences, the TSP is used to schedule a robot to drill holes in a circuit board (Ball and Magazine 1988), for genome sequencing (Johnson and Liu 2006), to design the layout of a satellite module (Sun and Teng 2003), optimal production sequencing (Jeong et al. 1997), as well as many other novel uses. Perhaps as important is that the TSP has become a fundamental platform for the study and assessment of general methods that can be applied to solve discrete optimization problems in different fields of research. There are several reasons for the TSP to play such an important role in combinational optimization. First, the TSP is a conceptually simple problem that turns out to be hard to solve because of being an NP-hard problem. Second, the TSP has no other additional constraints that are usually difficult to control in practice (Hoos and Stützle 2005). Finally, the TSP is the substructure of many other problems arising in real-life practical situations (Christofides 1979).

Many approaches have been developed to solve the TSP (Rego et al. 2011). Recently, ant colony algorithms have been proposed as a new heuristic approach for solving discrete optimization problems in general, and the TSP in particular. These algorithms simulate the collective intelligence of ants living in the same colony, particularly their foraging behavior. Although the Ant Colony System (ACS) and its variants have been successfully applied to various optimization fields, it does come with major drawbacks. Most significantly, when the number of cities to visit becomes large, huge computation time and failure to find an optimal solution at convergence become problematic. This chapter aims at improving the performance of the ACS with respect to these two key considerations by proposing enhancements that take advantage of the known spatial structure of the set of cities to be visited. An overview of the TSP is provided in the second section, while the principles of the Ant Colony Algorithm are presented in the third section. The enhanced ACS algorithm, dubbed the Nearest Neighbor Ant Colony (NNAC) algorithm, is presented next. The implementation of the NNAC algorithm is then pursued on Oliver-30, a dataset commonly used in testing and benchmarking described in Whitley et al. (1989), as well as four other derivative datasets. This provides the test bed for comparing the original ACS and the new NNAC algorithm. Conclusions are drawn at the end of the chapter.

The Traveling Salesman Problem

The traveling salesman problem (TSP) was first treated mathematically in the special case of the so-called Hamiltonian circuit in the 1800s by the Irish mathematician William R. Hamilton and by the British mathematician Thomas P. Kirkman. The general form of the TSP was first discussed in a series of publications by K. Menger in the late 1920s and the computational complexity of solutions to the TSP was addressed by H. Whitney in a 1934 seminar talk at Princeton University (Flood 1956; Schrijver 2005).

The TSP is known to be a NP hard problem for which no known efficient algorithm exists. The computational time to solve NP hard problems increases exponentially with problem size. Compared with any polynomial time problem whose computational time increases as N^2, where N is the problem size, the NP hard problem increases as 2^N instead. The TSP is very difficult to solve optimally due to its combinatorial complexity.

Since Dantzig et al. (1954) introduced the technique of plane cutting in integer programming, several modern and high-performance techniques have been developed to find the exact TSP solution. These include the branch and cut method (Padberg and Rinaldi 1991; Applegate et al. 2006), the branch and bound method (Little et al. 1963; Held and Karp 1970) and a few others. With enhanced linear programming-based techniques and greater computing power, problems of increasingly large sizes have been solved optimally, from Dantzig et al.'s 49 German cities in 1954 to Applegate et al.'s 85,900-city instance solved by the CONCORDE branch-and-cut algorithm (Applegate et al. 2006).

Given the large computing time of exact algorithms, practical TSP solution procedures are necessarily heuristic (Curtin 2007). Various approximation methods, whose rate of growth of computation time is a low order polynomial in n, have been experimentally observed to perform well (Christofides 1979; Curtin et al. 2014). The nearest neighbor heuristic (Hoos and Stützle 2005) is a form of greedy algorithm. Other approaches include simulated annealing (Meer 2007), Tabu search (Gendreau et al. 1994), and the Lin-Kernihan (LK) heuristic (Lin and Kernihan, Lin and Kernighan 1973). The LK algorithm is a particularly prominent TSP solution method. It is a tour improvement method that attempts to improve a given graph by exchanging a set of edges in order to obtain an alternative graph of lower cost (Applegate et al. 2006). The basic concept of the LK algorithm is based on the 2-opt moves method (Flood 1956).

Several heuristic methods based on a biological metaphor hold great promises for solving combinatorial optimization problems of the complexity exhibited by the TSP. The main approaches here are the genetic algorithm (GA), Neural Networks (NN), Particle Swarm Optimization (PSO), and Ant Colony Optimization. Some solution strategies resort to hybrid approaches where multiple heuristics are integrated (see for instance Marinakis et al. 2010; He and Mo 2011). The well-known approximate search technique of genetic algorithm (GA) simulates evolutional mechanisms of genetic material such as selection, mutation and crossover to determine the better solution from its prior generation (Shengwu and Chengjun 2002). The GA has a better chance that some other algorithms to avoid getting trapped in the local minimum because the mutation operator alters one or more gene values in a chromosome and can result in entirely new genes being added to the gene pool; hence the algorithm has the opportunity to escape from the local minimum. Neural networks find an optimal TSP solution through learning by simulating the human nervous system. Unsupervised forms of neural networks such as Kohonen's self-organizing map or adaptive resonance theory (ART) (Goldstein 1990; Vishwanathan and Wunsch 2001; Mulder and Wunch 2003) iteratively fit a prototype tour to the set of nodes to the visited in the solution space. PSO has a population of candidate solutions (particles) that move in the search-space, guided by their own best known position as well as the entire swarm's best known position in this space (Cunkas and Ozsaglam 2009). Ant colony optimization is described in more detail in the following section.

Ant Colony Optimization

Ants are known to have developed evolved social strategies to efficiency find food supply and identify short paths between their nest and food sources. Inspired from the behavior of real ant colonies, Dorigo's (1992) Ant System (AS) is a heuristic method that simulates individual and collective behaviors of an ant colony. It is often regarded as one of the most advanced techniques for approximate optimization across diverse domains of application (Blum 2005). Since the early 1990s, the ant colony algorithm has been used to solve discrete optimization problems, such as the traveling salesman problem (Colorni et al. 1992; Dorigo et al. 1996; Dorigo and Gambardella 1997), vehicle routing problems (Reimann et al. 2004; Abousleiman et al. 2017), packet-switched communication network problems (Schoonderwoerd et al. 1997; Di Caro and Dorigo 1998), the network design problem (Poorzahedy and Abulghasemi 2005), land cover zoning and planning (Li et al. 2011), and others.

Rather simple principles of behavior can explain not only the ability of ants to find the shortest path but also the ability to adjust to changes in their decision environment. Each ant has an inclination for certain chemical compounds called pheronomes, deposited on trails by other ants. When ants arrive at decision nodes where they have to decide what direction to follow, they tend to follow the path that has been discovered by most ants, which means that stronger pheromone has been deposited on this path. A strong pheromone on the shorter path will obviously be created much faster than on longer paths. This will prompt an increasing number of ants to choose the shorter path until most ants have found the shortest path.

As a form of swarm optimization approach, ant colony heuristics assign a large number of virtual ant agents to the task of exploring many possible paths between cities. As it builds a tour, each ant selects the next city to visit through a stochastic rule that features the amount of virtual pheromone deposited on the edges. The ants explore untraversed edges and in doing so, they deposit pheromone on edges until their tour is completed (local trail updating). Once all the ants have completed the shortest tour, virtual pheromone is deposited on the complete route (global trail updating). The amount of pheromone is inversely proportional to the tour length.

The AS has been used in different optimization applications and results have shown this algorithm to be capable of finding optimal solutions. In addition, AS is easy to understand, to program, and to combine with other heuristic algorithms. However, is suffers from serious drawbacks. Most notably, it consumes large computation time and gets stuck on local minima rather easily. The algorithm routinely takes at least 1–2000 iterations of all virtual ants through a solution search to find an optimal result and, even then, the algorithm could still have reached stagnation behavior (local minimum) without finding a global optimum.

Different strategies have been proposed to remedy this deficiency. For instance, Dorigo et al. (1996) introduced the Elitist Ant System (EAS) in which improvement is achieved by providing additional reinforcement to the edges belonging to the best tour found since the start of the algorithm. Stützle and Hoos (1997) developed a Max-Min ant system which imposes constraints on the updating of pheromone laid on the trail in order to solve the stagnation situation: ants can only deposit pheromone on the edges which belong to the tour that outperforms others in or up to the current iteration. Wu et al. (Wu et al. 1999) applied mutation features to the ant colony algorithm in order to prevent the early stagnation behavior and to decrease the computational time. In the Rank-based Ant System (Bullnheimer et al. 1999) the amount of pheromone deposited on the edges of a tour depends on how well this tour ranks against others. Zhu and Yang (2004) proposed a dynamic pheromone updating approach, which ensures that every ant's contribution can be efficiently adopted during the search phase. Dorigo and Gambardella's (1997) Ant Colony System (ACS) differs from the AS by granting virtual ants more intelligent ability to not only exploit learned knowledge but also of exploring tours other than good tours identified hitherto in the algorithm.

The search strategy of ACS and other ant colony algorithms involves the maintenance of a candidate list of all nodes that remain to be visited. Virtual ants search every possible node on this list for their next position, whether this is likely to be an efficient move or not. This strategy causes computation time to be wasted on searching inefficient candidates. Inefficient candidates are those nodes selected by ant agents at their current position that will, by no means, become optimal solutions. Furthermore, because pheromone is also deposited when inefficient nodes are visited, the probability that other ants choose the same route is raised, which may induce the algorithm to step into a stagnation situation without finding the best route, or lead to search results that are randomly scattered because ant agents are misled by the pheromone.

The Nearest Neighbor Ant Colony Algorithm

The Principle

As mentioned previously, the ant colony algorithms search every possible node listed in the candidate list. Thus, for a 30-node problem, 30! possible combinations are searched in order to find an optimal solution. In fact, most of the computation time involved in finding a solution may be wasted on node combinations that are not efficient. The Nearest Neighbor Ant Colony (NNAC) algorithm is proposed to improve computation time that is wasted on searching the inefficient combinations and to avoid early stagnation.

Careful observation of the Traveling Salesman Problem and of its solutions in the Euclidean plane indicates that optimal solutions commonly share a few fundamental properties. These properties are as follows:

1. Every node is linked by a solution edge to one of its nearest neighbors.
2. Outlier nodes, which are relatively far away from any others in the solution space, are not placed at the end of the visiting node sequence.
3. No two edges on a solution tour intersect, thus creating a planar solution.

These fundamental properties are exploited here in order to enhance the efficiency of ant colony algorithms. The proposed Nearest Neighbor Ant Colony (NNAC) algorithm incorporates a new searching strategy for the TSP that takes into account the spatial arrangement of cities across the study region. The algorithm responds to the spatial clumpiness of cities in the search for better paths traversing and connecting cities. It modifies the ant colony system to exhibit the properties described above so as to converge faster and thus reduce computing time. In addition, the NNAC implements a mutation function (Wu et al. Wu et al. 1999) that is intended to minimize the chance of stagnation on a local optimum. Some similar considerations drawn from local measurement of nearest neighbors were introduced by earlier researchers, in particular Doerner et al. (2002), Wetcharaporn et al. (2006), and Qi (2007). The novel structures of the proposed algorithm are discussed in more detail in the rest of this section.

The Nearest Neighbor Searching Strategy

The ACS algorithms sequentially search through many possible solutions (although not necessarily all of them); however, most of these solutions can readily be dismissed for being obviously suboptimal. In line with the first solution principle introduced above, the proposed NNAC algorithm focuses on nodes within a certain distance threshold r from current node i, instead of searching all nodes in the candidate list as a TSP tour is being built. The nearest candidate list, NNAC-candi,

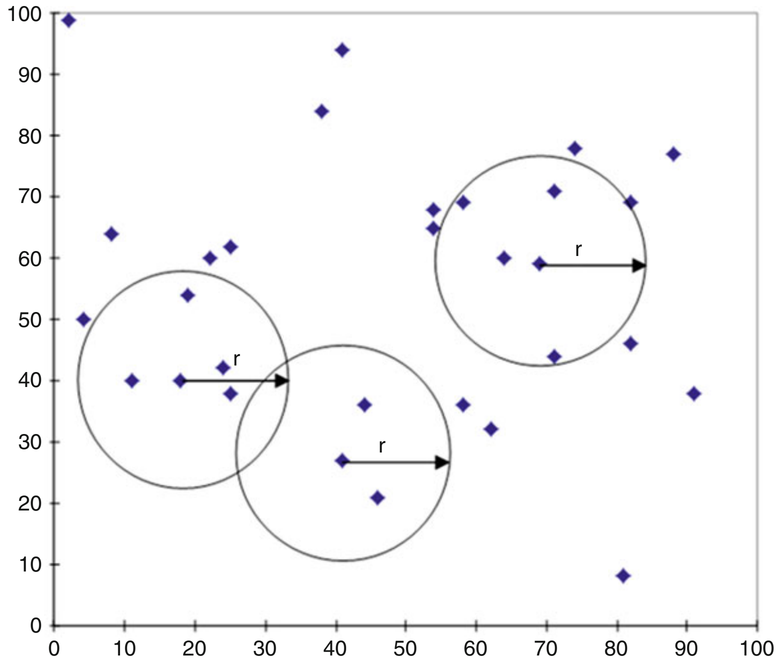

Fig. 1 The NNAC searching strategy

formed of nodes within the preset threshold eliminates most inferior nodes, which can reduce computation time to a great degree.

In contrast to the ACS algorithms that save every node that has never been visited to the candidate list, the NNAC-candi list of the Nearest Neighbor Ant Colony (NNAC) only saves a limited number of potential nodes which are located within a certain distance r to the ant's current location, as depicted in Fig. 1. It is a dynamic concept as it needs to be recomposed after each move on the tour by identifying all nodes encompassed by the moving circular window (buffer) of radius r. Practically, a sorted list of neighbors for all nodes can be computed in a pre-processing step and only those that are not tabu must be considered and placed in the NNAC-candi list. This search strategy takes into consideration the spatial process that generated the nodes to be visited. As a result, the number of nodes on an NNAC-candi list depends on the spatial structure of nodes in the vicinity of the current node: a local cluster produces a long list, while a sparse layout of competing and mutually avoiding nodes yields a short list. The search radius r is a critical parameter whose value directly impacts on the computation time as well as the optimality of the solution. The conventional candidate list is a limit case of the NNAC-candi list. The probability of choosing next node j from the NNAC-candi list, given the current node, follows the original ACS approach. It should be noted that parameter r must

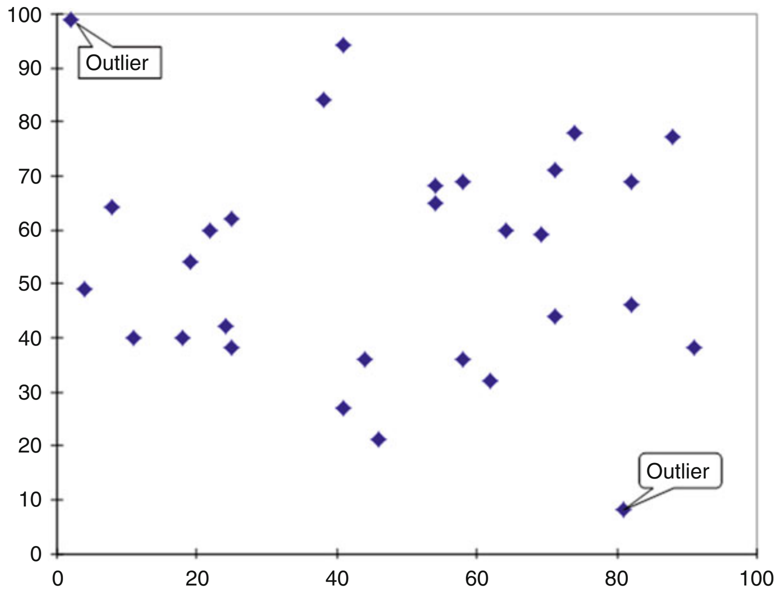

Fig. 2 Outlier nodes

be carefully selected to avoid: too small a value may exclude an optimal or near-optimal solution as the NNAC-candi list may be very short (and possibly empty), while a large r results in no computational savings.

The Problem of Outlier Nodes

Let us consider the nodes that are relatively far away from any others. The probability of selecting any of these outliers (Fig. 2) until close to the end of a tour is rather low since the distance between pairs of nodes is a controlling parameter of the probability function that drives the ACS algorithm. Therefore, in solutions produced by an ACS algorithm, outliers are usually listed at the end of the visiting list, which is at variance with known optimal solutions (second principle). Hence, whether an outlier is correctly placed in the visiting nodes sequence or not during the searching procedure is a critical criterion for finding an optimal solution. In order to enhance the proposed algorithm so that outliers are not ignored, an outlier-first searching strategy is deployed. Instead of randomly selecting a start node as in ACS, outliers are here given higher priority of being processed.

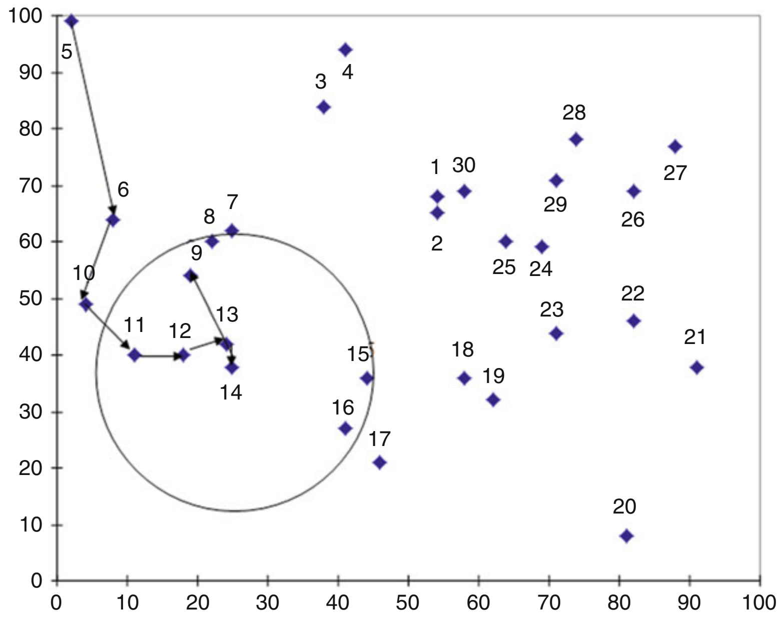

Fig. 3 Planar searching strategy

Planar Searching Strategy

Let us consider the scenario depicted by Fig. 3, where an ant agent starts from node 5. According to the NNAC, this ant chooses the sequence of nodes 6, 10, 11, 12, 13, 14, and so on. Up to this point, the searching sequence has followed a nearest neighbor searching strategy, according to which each new edge connects a node to its nearest neighbor. Let us now look at the NNAC-candi list of an agent arrived at node 14; this list includes nodes 9, 15, 16, and 8. According to the probability function controlling edge formation, the ant agent chooses node 9 as its next destination. This selection satisfies the searching strategy rule that the node is within searching distance r. However, the final result will not be the optimal route because it violates a fundamental property of TSP solutions, namely that segments $\overline{12, 13}$ and $\overline{9, 14}$ intersect. In order to reduce the likelihood that this property is violated, an improved pheromone updating rule is proposed.

Two pheromone updating strategies are applied in the proposed algorithm to represent the reinforcement mechanism that happens when agents explore routes and find good solutions. One is a local updating rule, while the other is an updating rule that takes into account the best tour found by an agent during each iteration. The latter replaces the global updating rule used in the ACS algorithm. Both rules contribute to reinforcing the ant agents' learning from their prior action and to speeding the convergence of the algorithm, while preserving the desired properties of TSP solutions.

Local Updating Rule

In the NNAC algorithm, the pheromone density value on each segment is updated right after an ant agent passes through. However, instead of adding a constant amount of pheromone on each segment as in the ACS, the amount decreases according to how far the ant agent has traveled so far. It is consistent with the expectation that the significance of information for making an optimal decision does not remain constant as an agent progresses along the route: information is more important at an early stage than in later phases. The route choices made early in the search are more important than those made later because each selection has a compounding impact on subsequent selections. A correct decision at an early stage will lead to a higher chance that the ant will make correct decisions at a later stage. When a selection is made at a more advanced stage of the tour, the remaining choices are so limited that its impact on the entire optimization becomes relatively minor. Meanwhile, the amount of pheromone also depends on whether the ant's searching strategy violates optimal route properties. For instance, if an ant agent picks its next node in a way that violates the properties of route planarity, the amount of pheromone deposited on this segment is discounted.

The local pheromone updating rule for any segment $\overline{\text{ij}}$is mathematically expressed as:

$$\tau_{ij}(t) = (1 - \rho_1) \quad \tau_{ij}\,(t - 1) + \Delta\tau_{ij}(t) \tag{1}$$

where t is the time index, such that one and only one edge is traversed per time unit, and $0 < \rho_1 < 1$ is a user-defined coefficient representing the rate at which pheromone evaporates between consecutive time periods t-1 and t. The term $\Delta\tau_{ij}(t)$ controls the deposit of pheromone during time period t. It lets ant agents learn the best action to perform in each possible state; it is defined as

$$\begin{cases} \Delta\tau_{ij}(t) = \frac{\sum \tau_{kl}(t)}{L} & \text{if segment } \overline{\text{ij}} \in \mathit{planar} \\ \Delta\tau_{ij}(t) = \frac{\sum \tau_{kl}(t)}{L}\,\delta & \text{otherwise,} \end{cases} \tag{2}$$

where L is the cumulative distance from start node s to current node i, the $\sum \tau_{ij}$ is the accumulated pheromone density from start node s to current node i, and $0 < \delta < 1$ is the scalar penalty assigned for violating the optimal route properties.

Iteration-Best Updating Rule

The iteration-best path is the shortest route of m runs (m ants) on any given iteration. The iteration-best updating rule is performed after all ants have completed their tour in the current iteration so as to reinforce the pheromone density on the segments belonging to the iteration-best route. By laying pheromone on the shortest route only, the NNAC iteration-best updating rule embodies the principle of "winner take all", which is intended to provide advantage to those segments which are identified as being part of the best combination.

The iteration-best rule for updating the pheromone value on segment $\overline{\mathrm{ij}}$ is expressed mathematically as follows:

$$\tau_{ij}\left(t=n\right)=\left(1-\rho_2\right)\quad\tau_{ij}\left(t=1\right)+\Delta\tau_{ij} \tag{3}$$

where t is the time index, $0<\rho_2<1$ is a user-defined coefficient representing the rate at which pheromone evaporates between the start ($t = 1$) and the end ($t = n$) of the tour construction. The term $\Delta\tau_{ij}$ controls the deposit of pheromone on the best tout of the iteration, in inverse proportion of its length. It is defined as

$$\Delta\tau_{ij}=\begin{cases}\left(L_{iteration-best}\right)^{-1} & \text{if } \overline{\mathrm{ij}}\in \text{iteration best tour}\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

where $L_{iteration-best}$ is the length of the iteration's shortest tour.

Mutation

Like most heuristic methods, ant colony optimization has the disadvantage of sticking on local minima; as a result, it may not be able to find the globally optimal route. This situation can be avoided by randomly disturbing the pheromone density of certain segments by resetting or increasing their values. This behavior is similar to the mutation function in genetic algorithms (Lin et al. 1993). During the iteration-best pheromone updating phase, the mutation function randomly switches the order of two nodes on the iteration-best route, which will cause an incorrect update of the pheromone density and also give the NNAC algorithm the opportunity to find the real global shortest path, if it happens to have erred towards a local optimum.

Procedures of NNAC and Comparison with ACS

The NNAC heuristic involves the iteration through a number of steps, which parallel the original ACS algorithm. For sake of clarity, a run of the simulation is defined as a complete route performed by a single ant agent. An iteration of an ant-based algorithm encompasses all the routing decisions made by all m ants. Therefore, in an m-ant simulation, an iteration has m runs and an ant run consists of n edges connecting nodes to form an ant's tour. A detailed description of ACS is available in Dorigo and Stützle (2004).

1. Choosing a start node

Instead of randomly selecting a start node from the set of nodes to be visited on a run as the original ACS does, the NNAC gives outliers more chances to be selected as a start node. For each odd run number, the NNAC randomly selects a start node from the outlier node pool. Conversely, on even runs, the start node is randomly selected from dataset as the traditional ACS does.

2. List of nodes that remain to be visited: NNAC-candi versus candidate list

For each ant k at node i, a NNAC-candi list is created. Only nodes within distance r from node i can be listed:

$$NNAC - candi_k = (j_1, j_2, j_3 \ldots j_n),$$

where j_n is a node within distance r of node i that has not yet been visited. In the ACS algorithm, all nodes which have not been visited are listed in the candidate list:

$candi_k = (j_1, j_2, j_3 \ \ldots \ j_n)$, where j_n is a node that has not yet been visited.

3. Choosing next node j

The NNAC algorithm applies the same choice rule as the ACS for selecting the node to which an ant moves at a certain run, given the list of nodes that are yet to be visited. This rule is given by

$$j = \begin{cases} \arg\max \quad \underset{s \in NNAC - candi_k(t)}{s \in candi_k(t)} \quad \left\{ [\tau_{is}(t)] \, [\eta_{is}]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases} \tag{5}$$

where q is a uniformly distributed random number defined on [0, 1], q_0 is a pre-defined parameter ($0 \le q_0 \le 1$), and S is the node that achieves the highest value on the following stochastic function:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} [\eta_{ij}]^{\beta}}{\sum \quad u \in candi_k(t) \quad [\tau_{iu}(t)]^{\alpha} [\eta_{iu}]^{\beta}} \\ \quad u \in NNAC - tabu_k(t) \\ if\ j \in candi_k(t) \text{ for ACS}, j \in NNAC - candi_k(t) \text{ for NNAC} \\ \quad 0 \qquad \text{otherwise} \end{cases} \tag{6}$$

where $\tau_{ij}(t)$ is the density of pheromone on edge (i,j) during time period t, η_{ij} is the inverse value of distance between node i and node j, and α , β are user-defined parameters which represent the importance of pheromone and distance, respectively, in the selection of the next node to visit. The latter equation is in fact the node selection rule of the original AS heuristic.

4. Local pheromone updating rule

In the ACS and NNAC algorithms, the pheromone density on each segment is updated right after an ant agent passes through. However, while a constant amount of pheromone is added on each segment in the ACS, the amount decreases according to how far the ant agent has traveled so far. The NNAC local updating rule is given by Eqs. (1) and (2). In contrast, in the standard ACS heuristic, the latter is replaced by

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} \text{ if ant k uses path (i, j) in its tour} \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

where Q is a constant and L_k is the length of the tour of the kth ant starting from its start node i, visiting all nodes, and returning to start node i.

5. Iteration-Best and global pheromone updating rule

In addition to the local updating rule, the NNAC performs an iteration-best pheromone updating rule at the end of each iteration so as to reinforce the pheromone density on the segments belonging to the iteration-best route. This rule is given by (3)–(4). It supersedes the ACS global pheromone updating rule, which is triggered at the end of each iteration when an ant k has identified the shortest tour during the searching period. The global pheromone updating rule is given by following equations:

$$\tau_{ij}\,(t = n) = (1 - \rho) \quad \tau_{ij}\,(t = 1) + \Delta\tau_{ij} \tag{8}$$

where

$$\Delta\tau_{ij} = \begin{cases} \left(L_{gb}\right)^{-1} & \text{if (i, j)} \in \text{global best tour} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and ρ is the pheromone decay parameter and $L_{gb}$is the length of the best of all the tours produced by all m agents since the beginning of the iteration.

6. Mutation

If the iteration-best route returns the same value for a number X of consecutive iterations, two nodes in the iteration-best visiting list are randomly switched and the pheromone values are updated accordingly. Number X is a user-defined constant that controls the rate of mutation by setting the permissible number of identical consecutive solutions. No mutation is allowed in the ACS heuristic.

7. Additional iterations

Steps 1 through 6 are repeated for a predetermined number of iterations and the shortest routes obtained in each iteration are then compared to produce the TSP solution.

Experimentation Results

In this section, we implement and test the NNAC heuristic on Oliver-30 (Whitley et al. 1989), a 30-city test problem commonly used in the literature for validation and benchmarking purposes. The NNAC heuristic is also tested on four other randomly generated 30-node datasets for consistency analysis. The experimentation results and the analysis of consistency of applying the NNAC algorithm on different datasets are reported below.

NNAC Performance on Five Datasets

Ant population size is set at 6 for all of the following experiments.[1] The outlier nodes are identified by visually examining the different datasets. In addition, good values of parameters α and β are determined by testing multiple combinations, the best of which is reported below for each data set. The other user-defined parameters are set as follows:

$$\text{pheromone decay } (\rho) = 0.5$$

$$\text{segment intersected penalty } (\delta) = 0.7$$

$$q_0 = 0.5.$$

Figure 4 depicts the results of one out of 50 trials with 100 iterations conducted on each test dataset. The left-hand side of each panel shows a plot of the length of the best route found by ant agents in each iteration of the depicted trial; the right-hand side maps the route that has the shortest length on all 100 iterations. As the results on the five 30-node datasets indicate, the NNAC generates TSP routes that are reasonable. Furthermore, the algorithm rather quickly converges to minimum-length routes that dominate the solutions after 50 iterations or so.

[1]Our experiments reveal that smaller population sizes degrade the performance of the solution algorithm. Conversely, large sizes increase computational time with no performance benefits.

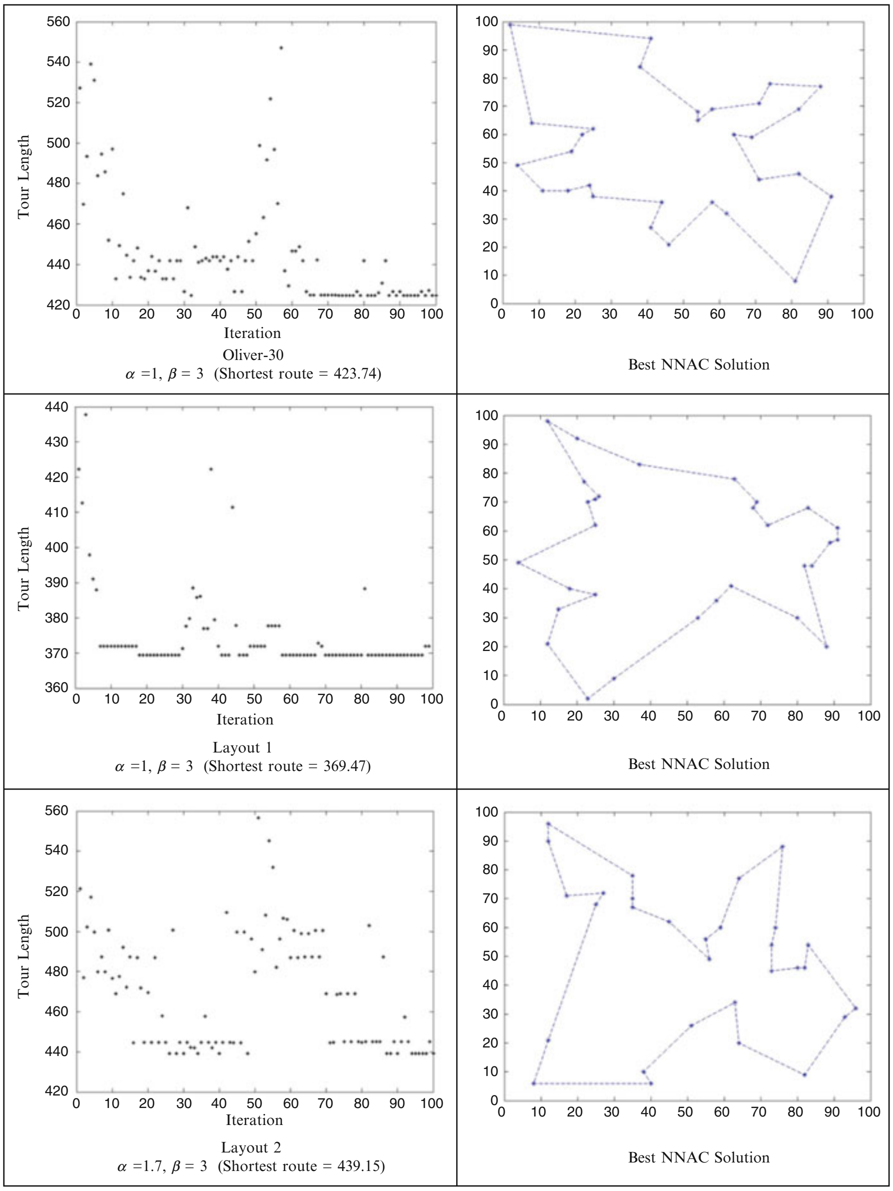

Fig. 4 NNAC results for five different city layouts

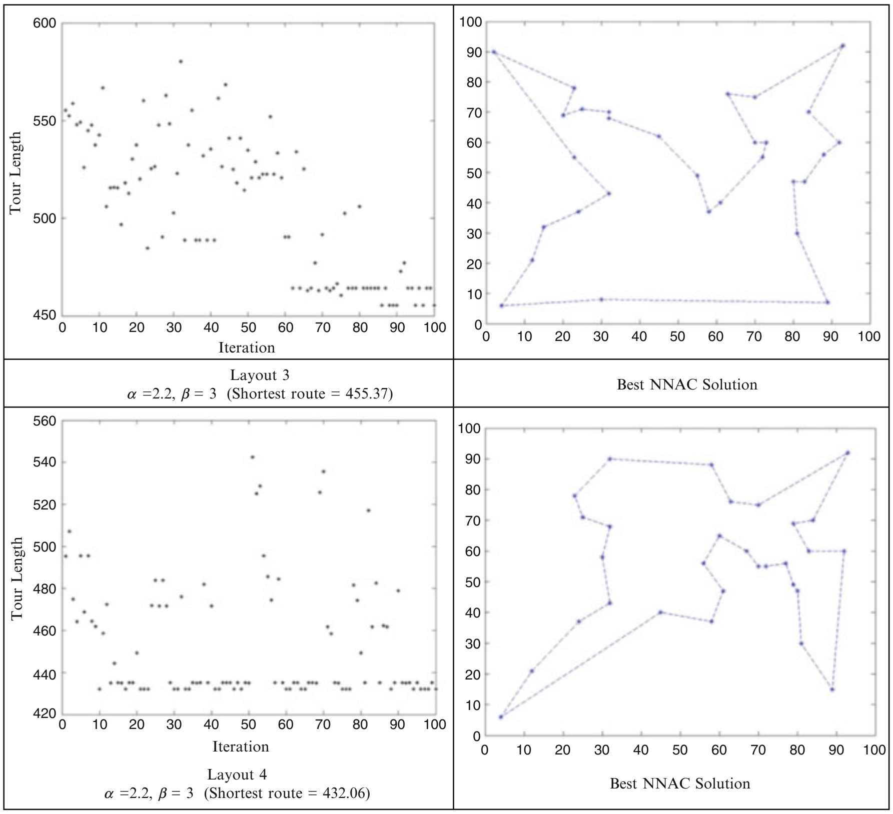

Fig. 4 (continued)

Consistency and Validation of NNAC Solutions

The Oliver-30 and four different 30-node layouts are also used to test the consistency of NNAC solutions. Results are obtained by running the NNAC algorithm on 50 trials; each trial includes 100 iterations with six ant agents. Table 1 reports the mean, standard deviation, and minimum of the best routes obtained over 50 trials. The average error is computed as the ratio between the mean of 50 best tour lengths and the length of the shortest tour. As shown in Table 1, the average error and the standard deviation are consistently small, which indicates that the performance of the NNAC algorithm is very consistent across all five test datasets.

While the NNAC heuristic is found to produce consistent TSP solutions, it is a worthy solution method only if it can be validated against known solutions. The best NNAC solution on Oliver-30 depicted in Fig. 4 (with a length of 423.74) is also found to be the best known solution of this test dataset (Whitley et al. 1989). No similar statement can be made for the other four test layouts as there are no known

Table 1 Summary statistics of NNAC best tour lengths for five test data sets

	Best tour length			
	Mean	Standard deviation	Minimum (best solution)	Average error (%)
Oliver-30	425.46	2.44	423.74	0.4
Layout 1	369.47	0.00	369.47	0.0
Layout 2	444.97	5.77	439.15	1.3
Layout 3	459.54	4.50	455.37	0.9
Layout 4	438.32	7.31	432.06	1.5

best solutions for these data sets. However, the best solutions attained in our tests can be benchmarked against known fundamental properties of optimal solutions reported in section "The Principle". All best solutions generated in our experiments meet the optimal properties mentioned which are:

1. Every node is linked by a solution edge to one of its nearest neighbors;
2. Outliers are correctly selected: the to-nodes of outliers are limited to their nearer neighbors;
3. No two link segments intersect on the route.

Furthermore, the NNAC heuristic is found to converge to a best solution exhibiting these desirable properties in 100 iterations or less (See Fig. 4). This represents a significant improvement in efficiency in solving the TSP with ant colony algorithms. In fact, the NNAC heuristic outranks the efficiency of the ACS algorithms by a large margin, since the latter typically need thousands of iterations to find the best solution. It has already been noted earlier that the standard deviations and average error percentages yielded by the NNAC algorithm on various data sets are small, which indicates that the solutions generated by the NNAC algorithm tightly cluster around the best value. These two considerations lead us to conclude that the NNAC heuristic is a reliable and robust solution approach to the TSP.

Experiment Comparison

The performance of the NNAC algorithm can also be assessed by comparing its solutions to those of the ACS on the same dataset. Dorigo and Gambardella (1997) conducted Ant Colony System (ACS) experiments with three different specifications of the local pheromone updating rule (1) on the Oliver-30 dataset. Their TSP solutions are reported here. The three ACS variants differ by the term $\Delta\tau_{ij}$ that controls the deposit of new pheromone during each time period. Specifically, in the simple ACS, $\Delta\tau_{ij} = \tau_0$, where τ_0 is a constant; the ant-Q updating rule uses $\Delta\tau_{ij} = \gamma \max \tau_{ij}$, where γ is a user-defined parameter; finally $\Delta\tau_{ij} = 0$ is assumed in a naïve ACS heuristic.

Table 2 Experiment comparison on Oliver-30

Heuristic	Tour length		
	Average	Standard deviation	Minimum
NNAC	**425.46**	**1.44**	**423.74**
Simple ACS	424.74	2.83	423.74
Ant-Q	424.70	2.00	423.74
Naïve ACS with $\Delta\tau(r,s) = 0$	427.52	5.21	423.74

The ACS solutions with the three different pheromone updating rules serve to benchmark the NNAC solutions. The local updating and global updating rules in the NNAC are described by Eqs. (1–2) and (3–4), respectively. Table 2 shows the results obtained from Dorigo and Gambardella (1997) and NNAC.

According to the results shown in Table 2, the NNAC heuristic and the ACS with all three alternate specifications of $\Delta\tau_{ij}$can find the best TSP solution with a route length of 423.74. Although the simple ACS and Ant-Q approaches have better performance in term of average tour length, the lower standard deviation of the NNAC indicates that the NNAC algorithm more consistently finds a close to optimal solution than the variants of the ACS algorithm.

It should be pointed out that experiments with NNAC involve 25,100-iteration trials with 6 ants. On the other hand, results reported for the three ACS variants were generated by 25 2500-iteration trials with 10 ants. Therefore, the total number of ant runs in the latter experiments is 625,000, which contrasts with a mere 15,000 runs of the NNAC. The NNAC uses 2.4% of the ant runs of more traditional ACS algorithms. Remarkably, not only does the NNAC generate TSP solutions with less variance than the ACS algorithm, it also does so in a fraction of the computing time.

Conclusions and Future Enhancements

While ant colony heuristics have been successfully applied to various optimization problems such as the TSP, they suffer from huge computation time and from a tendency to settle on local optima at convergence. In this chapter, we have proposed several enhancements to the conventional ACS algorithms. The proposed NNAC algorithm capitalizes on the optimal route properties to reduce computing time without sacrificing on the optimality properties of the solutions. The performance of this algorithm was shown to be better than that of the ACS algorithm on computing time by eliminating inefficient routes in advance. Instead of searching every possible node, the NNAC searching strategy exploits the spatial structure existing among cities to be visited by focusing on nodes within a certain radius from the current node. Thus, the time saving comes from narrowing down the possible solutions from 30! to less than $5^{25} * 4!$ in a 30-node problem.

The NNAC also brings to bear prior knowledge on the spatial distribution of nodes in handling nodes which are relatively far away from the other nodes. These spatial outliers are largely ignored during the searching phase of ACS, which causes pheromone density to be inadequately updated and computing time to be wasted. In the NNAC algorithm, the randomly selected start node strategy is replaced by a conditional outlier-first strategy. The advantages of the conditional outlier-first strategy are:

1. An increase of the reliability of pheromone update rules;
2. A Reduction of the information noise generated by ignored outliers;
3. A reduction of the computing time.

Two enhanced pheromone updating rules are implemented in the NNAC. The local pheromone updating rule follows the traditional ACS approach but gives more intelligence while updating the pheromone density. In lieu of increasing the pheromone density uniformly, the NNAC local updating rule adds an amount according to the distance that an ant has traveled, which reinforces the importance of information received in the early searching stage. The NNAC local updating rule also assigns a penalty to non-planar link segments. The iteration-best updating rule, on the other hand, reinforces the information on the most efficient solution in each iteration, which reduces the probability of inefficient nodes being selected in the remaining iterations.

A mutation function is also implemented in the NNAC algorithm. The mutation function prevents the algorithm from being limited to a local minimum and falling in convergence without finding the global minimum. The mutation function in the NNAC is executed in the global pheromone update phase. By doing so, it gives the NNAC the chance to escape from a possible local minimum and to find the real global minimum value.

Currently the NNAC algorithm focuses on problem of small to moderate sizes. The NNAC features several solution strategies designed to consistently generate optimal solution properties. It provides an approach for eliminating inefficient node combinations in advance so as to reduce the computing time on the TSP problem. Thorough testing of the heuristic is in order to identifying best performing combination of parameters. Further parameter sensitivity analysis is needed to establish rules of thumb for the selection of appropriate parametric specifications. It is also anticipated that further enhancement of the efficient of the NNAC solution approach can be achieved by capitalizing to a greater extent on the spatial structure of nodes. Particularly, knowledge on the presence of local clusters of nodes can be exploited to avoid searching along dominated solution paths. It is our contention that the articulation of techniques of spatial point pattern analysis with ant colony optimization principles will lead to solution approaches suitable for large-scale TSP problems. Also, the algorithm should be tested and customized to problems on networks, which have become rather standard in an era where much geospatial data are available on transportation infrastructure and their operational properties, such as speed and capacity.

References

Abousleiman, R., Rawashdeh, O., & Boimer, R. (2017). Electric vehicles energy efficient routing using ant Colony optimization. *SAE International Journal of Alternative Powertrains, 6*(1).

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study*. Princeton: Princeton University Press.

Ball, M. O., & Magazine, M. J. (1988). Sequencing of insertions in printed circuit board assembly. *Operations Research, 36*, 192–201.

Blum, C. (2005). Ant Colony optimization: Introduction and recent trends. *Physics of Life Reviews, 2*, 353–373.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). A new rank based version of the ant system: A computational study. *Central European Journal of Operations Research, 7*, 25–38.

Christofides, N. (1979). The traveling salesman problem. In N. Christophides, A. Mingozzi, P. Toth, & C. Sandi (Eds.), *Combinatorial Optimization* (pp. 131–149). New York: Wiley.

Colorni, A., Dorigo, M., & Maniezzo, V. (1992). An investigation of some properties of an ant algorithm. In *Parallel problem solving from nature conference (PPSN 92)* (pp. 509–520). New York.

Cunkas, M., & Ozsaglam, M. Y. (2009). A comparative study on particle swarm optimization and genetic algorithms for traveling salesman problems. *Cybernetics and Systems, 40*, 490–507.

Curtin, K. (2007). Network analysis in geographic information science: Review, assessment, and projections. *Cartography and Geographic Information Science, 34*, 103–111.

Curtin, K., Voicu, G., Rice, M. T., & Stefanides, A. (2014). A comparative analysis of traveling salesman solutions from geographic information systems. *Transactions in GIS, 18*, 286–301.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling salesman problem. *Operations Research, 2*, 393–410.

Doerner, K. F., Gronalt, M., Hartl, R. F., Reimann, M., Strauss, C., & Stummer, M. (2002). Savings ants for the vehicle routing problem. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, & G. R. Raidl (Eds.), *Applications of evolutionary computing* (pp. 11–20). Berlin/Heidelberg: Springer LNCS 2279.

Di Caro, G., & Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communication networks. *Journal of Artificial Intelligence Research, 9*, 317–365.

Dorigo, M. (1992). *Optimization, learning and natural algorithms*. Unpublished Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M., & Gambardella, L. (1997). Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*, 53–66.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a Colony of cooperating agents. *IEEE Transactions on System, Man, and Cybernetics—Part B: Cybernetics, 26*, 29–41.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge, MA: MIT Press.

Exnar, P., & Machac, O. (2011). The travelling salesman problem and its application in logistic. *WEAS Transactions on Business and Economics, 18*, 163–173.

Flood, M. M. (1956). The traveling salesman problem. *Operations Research, 4*, 61–75.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A Tabu search heuristic for the vehicle routing problem. *Management Science, 40*, 1276–1290.

Goldstein M (1990) Self-organizing feature maps for the multiple traveling salesman problem (MTSP). In: *Proceedings IEEE international conference on neural networks*, Paris, pp. 258–261.

He, X. B., & Mo, Y. W. (2011). Solving the TSP by simulated annealing genetic algorithm based on Google maps JavaScript API. *Advanced Materials Research, 201–203*, 733–737.

Held, M., & Karp, R. M. (1970). The traveling salesman problem and minimum spanning trees. *Operations Research, 18*, 1138–1162.

Hoos, H. H., & Stützle, T. (2005). *Stochastic local search: Foundations and applications*. San Francisco: Morgan Kaufmann.

Jeong, E. Y., SC, O., Yeo, Y. K., Chang, K. S., Chang, J. Y., & Kim, K. S. (1997). Application of traveling salesman problem (TSP) for decision of optimal production sequence. *Korean Journal of Chemical Engineering, 14*, 416–421.

Johnson, O., & Liu, J. (2006). A traveling salesman approach for predicting protein functions. *Source Code for Biology and Medicine, 1*, 3.

Li, X., Lao, C. H., Liu, X. P., & Chen, Y. M. (2011). Coupling urban cellular automata with ant Colony optimization for zoning protected natural areas under a changing landscape. *International Journal of Geographical Information Science, 25*, 575–593.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research, 21*, 498–516.

Lin, F. T., Kao, C. Y., & Hsu, C. C. (1993). Applying the genetic approach to simulated annealing in solving some NP-hard problems. *IEEE Transactions on Systems, Man and Cybernetics, 23*, 1752–1767.

Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research, 11*, 972–989.

Marinakis, Y., Marinaki, M., & Dounias, G. (2010). A hybrid particle smarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence, 23*, 463–472.

Meer, K. (2007). Simulated annealing versus metropolis for a TSP instance. *Information Processing Letters, 104*, 216–219.

Mulder, S. A., & Wunch, D. C. (2003). Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks. *Neural Networks, 16*, 827–832.

Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review, 33*, 60–100.

Poorzahedy, H., & Abulghasemi, F. (2005). Application of ant system to network design problem. *Transportation, 32*, 251–273.

Qi, C. (2007). An ant colony system hybridized with randomized algorithm for TSP. In: *Eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing* (SNPD 2007), pp. 461–465.

Rego, C., Gamboa, D., Glover, F., & Osterman, C. (2011). Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research, 211*, 427–441.

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problems. *Computers & Operations Research, 31*, 563–591.

Schoonderwoerd, R., Holland, O., Bruten, J., & Rothkrantz, L. (1997). Ant-based load balancing in telecommunications networks. *Adaptive Behavior, 5*, 169–207.

Schrijver, A. (2005). On the history of combinatorial optimization (till 1960). In K. Aardal, G. L. Nemhauser, & R. Weismantel (Eds.), *Handbook of discrete optimization* (pp. 1–68). Amsterdam: Elsevier.

Shengwu, X., & Chengjun, L. (2002). A distributed genetic algorithm to TSP. In: *Proceedings of the 4th world congress on intelligent control and automation* (Vol. 3, pp. 1827–1830).

Song, S. H., Lee, K. S., & Kim, G. S. (2002). A practical approach to solving a newspaper logistics problem using a digital map. *Computers and Industrial Engineering, 43*, 315–330.

Spada, M., Bierlaire, M., & Liebling, T. M. (2005). Decision-aiding methodology for the school bus routing and scheduling problem. *Transportation Science, 39*, 477–490.

Sun, Z. G., & Teng, H. F. (2003). Optimal layout design of a Satellite Module. *Engineering Optimization, 35*, 513–529.

Stützle, T., & Hoos, H. H. (1997). MAX-MIN ant system and local search for the traveling salesman problem. In: *IEEE Int'l conference on evolutionary computation* (pp. 309–314). IEEE Press.

Thill, J. C., & Thomas, I. (1991). Towards conceptualizing trip-chaining behavior: A review. *Geographical Analysis, 19*, 1–17.

Vishwanathan, N., & Wunsch, D. C. II. (2001). ART/SOFM: A hybrid approach to the TSP. In: *Proceedings of neural networks* (Vol. 4, IJCNN '01, pp. 2554–2557). International Joint Conference, Washington, DC.

Wetcharaporn, W., Chaiyaratana, N., & Tongsima, S. (2006). DNA fragment assembly: An ant colony system approach. In F. Rothlauf et al. (Eds.), *Applications of evolutionary computing. EvoWorkshops 2006. Lecture notes in computer science* (Vol. 3907, pp. 231–242). Berlin/Heidelberg: Springer.

Weigel, D., & Cao, B. (1999). Applying GIS and OR techniques to solve Sears technician-dispatching and home delivery problems. *Interfaces, 29*, 112–130.

Whitley, D., Starkweather, T., & Fuquay, D. (1989). Scheduling problem and traveling salesman: The genetic edge recombination operator. In: *Proceedings of the third international conference on genetic algorithm*, Morgan Kaufmann, Palo Alto, CA, pp. 133–140.

Wu, Q. H., Zhang, J. H., & XH, X. (1999). An ant colony algorithm with mutation features. *Journal of Computer Research and Development, 36*, 1240–1245.

Zhu, Q. B., & Yang, Z. J. (2004). An ant colony optimization algorithm based on mutation and dynamic pheromone updating. *Journal of Software, 5*, 185–192.