Journal of the Operational Research Society (2014) 65, 1423–1436

www.palgrave-journals.com/jors/

The travelling maintainer problem: integration of condition-based maintenance with the travelling salesman problem

Fatih Camci*

Antalya International University, Döşemealtı/ANTALYA, Turkey

The Travelling Salesman Problem (TSP) is one of the most studied problems in the literature due to its applicability to a large number of real cases. Most variants of the TSP consider total distance travelled. This paper presents a new generalised formulation of the TSP that aims to minimise the sum of functions of latencies to cities, rather than total distance travelled. Then, a new problem that uses a special function using the latency as input is presented, called the Travelling Maintainer Problem (TMP). The TMP integrates the output of prognostics in Condition-based Maintenance (CBM) with the TSP. CBM aims to minimise the failure and maintenance cost by identifying and predicting upcoming failures through the analysis of sensory information collected in real-time. Maintenance scheduling is performed using the predicted failure information obtained from the CBM. When the systems to be maintained are geographically distributed, maintenance scheduling requires integrated analysis of travel times and their effects on the failure progression in systems. This paper also presents Genetic Algorithm and Particle Swarm Optimisation-based solutions and their comparisons for the TMP on a case study.

Journal of the Operational Research Society (2014) **65**(9), 1423–1436. doi:10.1057/jors.2013.88
Published online 7 August 2013

Keywords: Travelling Repairman Problem; Condition-based Maintenance; maintenance scheduling; prognostics; delivery-man problem; minimum latency problem

1. Introduction

This paper aims to define a new variant of Travelling Salesman Problem (TSP) for operations research community to address the non-linear effect of the latency to the cities. Latency is the travel time spent to reach the city from the starting point. The presented problem aims to find the best route among cities to minimise the sum of the travel and consequential non-linear cost of the latency to the cities. A simple example involves multiple cities, each of which has a system to be maintained, and a maintainer who is responsible for the maintenance of all systems. Each system has a forecasted failure probability for a given period representing the non-linear effect of the latency. The later the maintainer is, the more likely the system will fail before he reaches to the destination. The goal is to find the best route for the maintainer that minimises the travel, maintenance, and expected failure cost for all cities.

TSP is one of the most widely studied combinatorial optimisation problems in existence and which aims to find the routing of a salesman who starts from an origin (ie a home location), visits a prescribed set of cities, and returns to the origin in such a way that the total travelled distance is minimised and each city is visited once. Although the TSP, which was defined as a mathematical problem in the 1930s, is NP-hard, it has become very popular due to its potential applications in many real-world problems, such as scheduling, logistics, microchip production, and DNA sequencing (Laporte and Palekar, 2002; Punnen, 2002; Fakcharoenphol *et al*, 2007; Erdoğan *et al*, 2007; Vansteenwegen *et al*, 2009; Balaprakash *et al*, 2010; Bontoux *et al*, 2010; Angelelli *et al*, 2011; Gouveia *et al*, 2011; Casazza *et al*, 2012; Erdoğan *et al*, 2012; Glomvik Rakke *et al*, 2012). A very large number of papers and books have been published on this problem (Punnen, 2002). Different versions of this problem have been defined for many different possible applications (Laporte and Palekar, 2002; Fakcharoenphol *et al*, 2007; Erdoğan *et al*, 2007; Vansteenwegen *et al*, 2009; Balaprakash *et al*, 2010; Bontoux *et al*, 2010; Angelelli *et al*, 2011; Gouveia *et al*, 2011; Casazza *et al*, 2012; Glomvik Rakke *et al*, 2012; Erdoğan *et al*, 2012). Most of the studies on variants of TSP deal with total distance travelled, not the latency to reach a city.

The TSP in its simplest form finds the order of a finite number of cities to be visited that will lead to a minimum total travelling distance. In the standard TSP problem, every city has to be visited exactly once, and no value is associated with the service. There exist many variants of the TSP in the literature. The TSP with Profits (Feillet *et al*, 2005), the Generalised TSP

*Correspondence: Fatih Camci, Industrial Engineering Department, Antalya International University, Üniversite Cad. No: 2 07190, Döşemealtı/ANTALYA, Turkey.

(GTSP), and the Travelling Repairman Problem (TRP) (also known as the delivery-man problem) (Fischetti *et al*, 1993) are examples of variants of the TSP.

The TSP with Profits selects cities depending on a profit value that is gained when the visit occurs. There are two criteria in this problem: the distance travelled and the gain obtained by visiting. These two criteria can be combined into one objective function, or one criterion might be used as a constraint. In the former case, the aim is to find a tour that minimises the total distance, also called the profitable tour problem (Dell'Amico *et al*, 1995). In the latter case, the aim becomes either to maximise the gain of visiting cities by constraining the total travelled distance, called the Orienteering Problem (Vansteenwegen *et al*, 2009) or to minimise the distance travelled to achieve a given profit, called the Prize Collecting TSP (Balas, 1995). The GTSP clusters the cities to be visited (Fischetti *et al*, 1993). When constrained to visit one city in each cluster, the problem becomes the Equality GTSP. The full summary of the variants of the TSP is outside the scope of this paper.

The *TRP* is different from the variants of the TSP mentioned above because the TRP takes latency into account. The latency to a city is simply added in the objective function of the TRP. The TRP minimises the total latency to the cities, which is not the case in TSP (Fischetti *et al*, 1993), Minimum Latency Problem (Blum *et al*, 1994), or the TSP with cumulative cost (Bianco *et al*, 1993). In *k*-TRP, *k* tours is the aim (Fakcharoenphol *et al*, 2007). The *k*-TRP with repair time accounts for the time to be spent during the visit to each city (Jothi and Raghavachari, 2007). The TRP can be defined mathematically as finding the $\pi_0\pi_1\pi_2 \ldots \pi_n$ sequence that minimises the equation below:

$$\text{Minimize } z = \sum_{i=1}^{n+1} \sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k} \tag{1}$$

where:

- $\pi_{n+1} = \pi_0$;
- $\sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k}$ is the latency of π_i (time of travel from π_0 to π_i);
- d_{ij} is the distance between locations i and j;
- π_0 is the starting location that may not have a system to maintain;
- π_i is the ith city to be visited;
- z is the objective function to be minimised.

Vehicle routing problem, a generalisation of TSP, aims to find the route to multiple cities by multiple vehicles (Yu and Liu, 2011; Nagarajan and Ravi, 2012). Asymmetric patrolling repairman is a variant of TRP (Das and Wortman, 1993). The problem presented here is different and uses the latency information as input for a pre-defined function.

This paper aims to present a new generalisation of TSP and to define a new problem called Travelling Maintainer Problem (TMP) based on the presented generalisation. Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) based solution to the defined problem is given and their application on a case study is discussed in the paper. The paper is organised as follows: Section 2 presents the new generalised TSP and TMP. Section 3 describes the GA and PSO-based solutions to the TMP. Section 4 discusses a case study with analysis and comparison of solutions. Section 5 concludes the paper.

2. Travelling Maintainer Problem (TMP)

In this section, the TMP is discussed after presenting the generalised formulation of the TSP. The new general formulation of the TSP is given in (2) and can be stated as the minimisation of the sum of functions that use the latency to the city as input. This formulation is important when the effect of latency is complex and cannot be represented as a simple sum. Note that the formulation will be equal to the standard TSP given in (1), if the term $f_{\pi_i}(\sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k})$ is replaced by $d_{\pi_{i-1}\pi_i}$ where $K = n + 1$ and $\pi_{n+1} = \pi_0$. Similarly, this formulation will also be equal to TRP given in (1), if the term $f_{\pi_i}\left(\sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k}\right)$ is replaced by $\sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k}$.

$$\text{Minimise } z = \sum_{i=1}^{k} f_{\pi_i}\left(\sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k}\right) \tag{2}$$

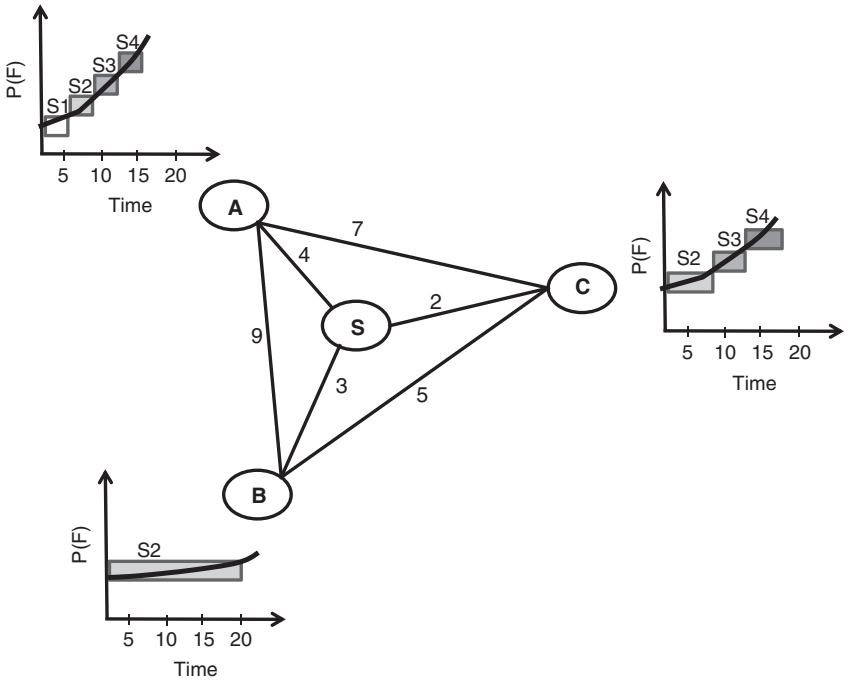
where:

f_{π_i} is a function that uses the latency as input for city π_i; and $\pi_{n+1} = \pi_0$.

The total number of visits is represented by K in (2), which may be different from the number of cities (n). Hence, it is possible that some cities may be visited more than once or not visited at all. The planned time period (T), which begins with the current time and ends with the last scheduled maintenance, is highly related with K. Setting one will define the other. Note also that the latency function (f_i) for each city might be different for each city because the effect of latency may be different for different cities. When the latency function is defined as the cost of failure, which is highly related to the time of maintenance, then the problem becomes the TMP.

In today's competitive industrial environment, maintenance has become very critical to achieve a high production rate, safety, and a low operation and support cost (Camci and Chinnam, 2010). Thus, electro-mechanical systems are being monitored real-time to identify impending failures and forecast the failure probabilities. The TMP is applicable when the systems are under real-time monitoring and located in various places and a maintainer needs to travel between these places for maintenance. As the latency increases, the probability of failure of systems located in all cities increases. The effect of maintenance latency in cities may be different based on the criticality of the city. The current and forecasted health status of cities (or systems in cities) may also be different. Thus, each city has a different function of latency that affects the objective function.

Condition-based Maintenance (CBM) is the common name used for real-time monitoring of the systems. Since CBM is not the main focus of this paper, readers are referred to other publications for further details (Camci and Chinnam,



Path	Travel	Visit Times				Failure Probabilities		
		π_1	π_2	π_3	π_4	π_1	π_2	π_3
S A B C S	4+9+5+2	4	13	18	20	S1	S2	S4
S A C B S	4+7+5+3	4	11	16	19	S1	S3	S2
S C A B S	2+7+9+3	2	9	16	21	S2	S2	S2
S C BA S	2+5+9+4	2	7	16	20	S2	S2	S4
S B C A S	3+5+7+4	3	8	15	19	S2	S2	S3
S B A C S	3+9+6+2	3	12	18	20	S2	S3	S4

Figure 1 Illustration of the TMP.

2006; Jardine *et al*, 2006; Camci, 2009a; Camci and Chinnam, 2010; Carr and Wang, 2011). Figure 1 illustrates the TMP. Each city has the forecasted failure probability of a system (eg, electromechanical system). The *x*-axis shows the time, whereas the *y*-axis shows the probability of failure. The current time is given as $x = 0$. As time passes, the probability increases. Node S in the middle represents the starting point. Nodes A, B, and C represent the cities with systems to be maintained. Each system has different forecasted failure probabilities as mentioned before. All possible visiting paths with visit times are given in the table. To demonstrate the TMP, failure probabilities are shown with states (ie, S1, S2, S3, and S4) affected by the visit times. Note that states are used for demonstration purposes and actual probability values will be used in the rest of the paper. Wind turbine farms located at various places in the sea, solar panels installed at various locations in the desert or railway switches located at various places in a city are examples of applications of the TMP.

The TMP aims to minimise the total cost by scheduling a maintenance (ie, visiting a city), which is constrained on the travel times between cities. The total cost consists of the expected failure, maintenance, and travel costs. Failure probabilities and maintenance costs vary for cities. For example, in Figure 1, the current failure probability in city A is low (S1) and is expected to increase rapidly (until S4). The current failure probability in city C is a moderate (S2) and is expected to increase rapidly (until S4). The current failure probability in city B is a moderate (S2) and is expected to increase very slowly (stay in S2). The TMP aims to find the best maintenance schedule that minimises the total cost.

The order of the maintenance schedule affects the time of the maintenance in the different cities. The maintenance start time affects the expected failure cost. Figure 2 displays the effect of maintenance on failure probability. When maintenance is not performed, the failure probability increases, as shown by the solid line. The effects of early and late maintenance are displayed by the dashed and dotted lines.

Maintenance reduces the failure probability, and the expected failure cost decreases with extra maintenance costs. Thus, excessive maintenance will not reduce the total cost. The latency function (f_{π_i}) defined in (2) includes maintenance $(C^M_{\pi_i})$ and the expected failure cost $(C^F_{\pi_i})$ as well as the travel cost $(C^T_{\pi_i})$.

$$f_{\pi_i} = C^F_{\pi_i}\left(\sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k}\right) + C^M_{\pi_i} + C^T_{\pi_i} \tag{3}$$

A failure causes two types of cost: direct failure cost and indirect failure cost. Direct failure cost (F_{π_i}) is the repair cost of the system after failure, whereas indirect failure cost is the cost

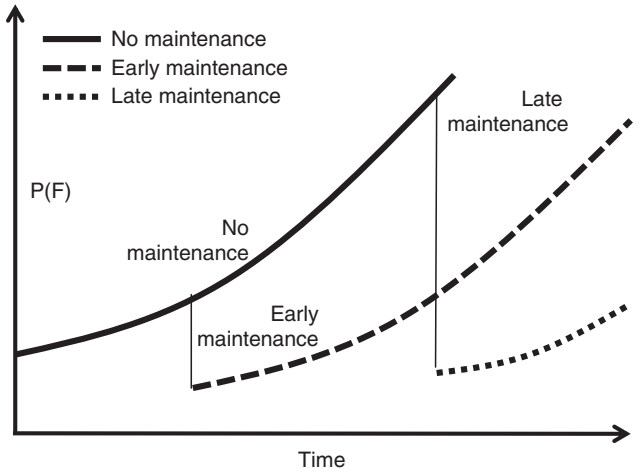


Figure 2 Effect of maintenance on failure probability.

when a system is down, such as the loss of production. The expected failure cost is the sum of the expected direct and indirect failure costs. The expected direct failure cost is calculated as the product of the cumulative failure probability (P_{π_i}) and the direct failure cost (F_{π_i}). The expected indirect failure cost is calculated as the product of the expected duration of downtime $(E[D_{\pi_i}])$ and the cost of having downtime, per unit time (δ_i). Note that P_{π_i} and D_{π_i} are variables related to the health of the system, whereas F_{π_i} and δ_{π_i} are predefined constants. The formulation of the expected failure calculation is given in (4).

$$C^F_{\pi_i}\left(\sum_{k=1}^{i} d_{\pi_{(k-1)}\pi_k}\right) = P_{\pi_i} \times F_{\pi_i} + E[D_{\pi_i}] \times \delta_{\pi_i} \tag{4}$$

The parameters F_{π_i} and δ_{π_i} are given as inputs. The expected downtime $(E[D_{\pi_i}])$ is calculated as the area under the failure probability curve, which is calculated as the trapezoidal numerical integration of the given curve, as shown in Figure 3 (the shaded area under the failure probability curves). P_{π_i} is the cumulative probability of failure within the planned time period ([0, T]), which is divided into sections according to a maintenance schedule. Note that planned time period ([0, T]) is highly related to K and can be calculated by a given K and sum of latencies to the cities. $P_{\pi_i}$is calculated by using the probability curves in all of the sections. For example, if no maintenance is scheduled in the period $[0, T]$, then only one section exists, and the failure probability is obtained from the prognostics module. If one maintenance is scheduled in $[0, T]$, then two periods are obtained, that is, before and after maintenance, as shown in Figure 3. Maintenance is scheduled at time MT_1, and the failure probability in $[0, T]$ is the union of the failure probabilities in $[0, MT_1]$ and $[MT_1, T]$ In other words, the probability of failure in $[0, T]$ is the sum of the probability of failure occurring before maintenance and the product of the probability of failure not occurring before maintenance and the probability of failure occurring after maintenance, as shown in (5). Note that probability values $(p_{\pi_i,1}, p_{\pi_i,2}, p_{\pi_i,3}, \ldots, p_{\pi_i,M+1})$ are obtained from failure probability curves supplied by

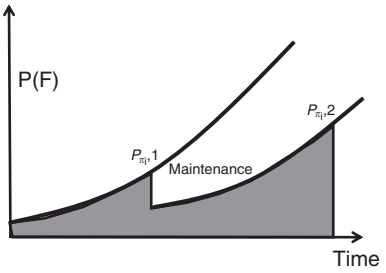


Figure 3 Sections obtained with maintenance and expected downtime.

prognostics module, which receives real-time sensory information and calculates the future failure probabilities and Remaining Useful Life (RUL). RUL is the time a mechanical system can function safely before failure occurs and can be represented as failure probabilities in future times. These numbers may be obtained from different curves representing different failure mechanisms in different cities and different last maintenance times.

$$p_{\pi_i} = p_{\pi_i,1} + (1 - p_{\pi_i,1}) \times p_{\pi_i,2} \tag{5}$$

where:

$p_{\pi_i,1}$ is the probability of failure before maintenance; and $p_{\pi_i,2}$ is the probability of failure after maintenance.

Note that the calculation above considers the probability of failure in the calculation of the expected cost of failure. Maintenance scheduling should be re-run if forecasted failure probabilities change in the future. When an incipient failure is detected (ie, the forecasted failure probability has increased) or a failure actually occurs (the failure probability becomes 1), then maintenance should be rescheduled with this new information. In other words, scheduled maintenance will be performed until the forecasted failure probabilities change or the scheduled maintenance is completed. Note that a system may continue to work with an incipient failure and that RUL is expected to be more accurate after an incipient failure is detected (Camci and Chinnam, 2010).

When there are M_{π_i} maintenance actions, $M_{\pi_i} + 1$ sections in the planned time period ([0, T]) are obtained. Note that T may either be given or calculated by given K. The failure probability at the mth maintenance is represented by $p_{\pi_i,m}$ ($p_{\pi_i,m+1}$ represents the failure probability at the end of the period). The formula for the cumulative probability calculation can be generalised in a recursive manner, as shown in (6):

$$P_{\pi_i,m+1} = P_{\pi_i,m} + (1 - P_{\pi_i,m}) \times p_{\pi_i,m+1} \tag{6}$$

where:

$m = 1, 2, \ldots, M$; $P_{\pi_i} = P_{\pi_i,M+1}$; $P_{\pi_i,1} = p_{\pi_i,1}$,

$P_{\pi_i,m}$ is the cumulative probability of failure before mth maintenance.

The maintenance cost $(C^M_{\pi_i})$ can be thought as a constant value for a city. However, if the system fails before the scheduled maintenance, then the maintenance will not be performed. Thus, the maintenance cost should be calculated as the expected maintenance cost by multiplying the fixed maintenance cost (γ_{π_i}) by the expected number of maintenance

actions for city i: $\left(\sum_{m=1}^{M\pi_i}(1-P_{\pi_i},m)\right)$:

$$C_{\pi_i}^{M}=\gamma_{\pi_i}\times\sum_{m=1}^{M_{\pi_i}}\left(1-P_{\pi_i,m}\right) \tag{7}$$

The travel cost $(C_{\pi_i}^{T})$ is the product of the cost of travelling per unit distance and the travelled distance, as shown in (8).

$$C_{\pi_i}^{T}=c_d d_{\pi_{i-1}\pi_i} \tag{8}$$

where
c_d is the travel cost of unit distance.

As a result, the objective function of the TMP is given as in (9) with given constants: $F_{\pi_i}, \delta_{\pi_i}, \gamma_{\pi_i}, K,$ and c_d.

$$\text{Minimize}\quad z=\sum_{i=1}^{K}\left(P_{\pi_i}\times F_{\pi_i}+E[D_{\pi_i}]\times\delta_{\pi_i}+\gamma_{\pi_i}\times\sum_{m=1}^{M_{\pi_i}}\left(1-P_{\pi_i,m}\right)+c_d d_{\pi_{i-1}\pi_i}\right) \tag{9}$$

Subject to:

$$0\leqslant P_{\pi_i}\leqslant 1,\qquad 1\leqslant i\leqslant K, \tag{10}$$

$$E[D_{\pi_i}]\geqslant 0,\qquad 1\leqslant i\leqslant K, \tag{11}$$

$$M_{\pi_i}\geqslant 0,\qquad 1\leqslant i\leqslant K, \tag{12}$$

$$0\leqslant P_{\pi_i,m}\leqslant 1\qquad 1\leqslant i\leqslant K \tag{13}$$

$$d_{\pi_{i-1}\pi_i}\geqslant 0,\qquad 1\leqslant i\leqslant K \tag{14}$$

Even though the TMP is defined based on the concept of maintenance of different systems located in various locations, fundamental objective function defined in Equation (2) is general and can be applied to different problems. The fundamental concept is the incorporation of the non-linear effect of latency to the cities. An example might involve a travelling salesman that sells a product whose value or amount to be purchased decreases non-linearly with the latency of delivery in different ways for cities.

3. GA/PSO-based solution method to the TMP

3.1. Genetic Algorithm (GA)

GA is a well-established method to approximate to the global optimum especially for non-linear problems. A GA does not guarantee obtaining a global optimal solution in finite number of iterations; however, GAs are a commonly used method for giving one of the best results possible. Because GAs are well established, the details of the GA methods will not be discussed in this paper.

GAs have been widely used for the TSP in the literature (Chatterjee *et al*, 1996; Moon *et al*, 2002; Bontoux *et al*, 2010), as well as for other tour-related problems (Dohn *et al*, 2011; Wen *et al*, 2011; Xiao *et al*, 2012). A good summary of GA-based methods for the TSP can be found in (Potvin, 1996).

Table 1 X chromosome example

City order	*Binary representation—Chromosome*
3 2 4 1 5	011 010 100 001 101
1 4 2 5 3	001 100 010 101 011

Application of the standard GA to the TSP has fundamental drawbacks, such as leading to infeasible tours with multiple or no visits to some cities. In addition, when the number of cities is not a power of two, GA operators on binary numbers may lead to some binary sequences that do not correspond to a city. Although the TSP requires one and only one visit to each city, this requirement is not true for the problem presented here (TMP), in which a city may not be visited or may be visited multiple times, as mentioned in the previous section. The only concern in applying standard GAs to the TMP is the possibility of having binary sequences that do not correspond to a city. This paper presents the standard GA representation for the TMP by penalising the solutions that include non-existing cities.

A number in binary format is assigned to each city. The length of the city number is $[\log_2 N]$, where N is the total number of cities, and is called a gene. The order of the cities represented by a sequence of binary variables (X) is called a chromosome as shown in Table 1. The length of the chromosome is $K\times[\log_2 N]$, where K is the number of visits.

This paper does not provide new GA operators (ie, crossover and mutation); rather, existing operators (ie, scattered crossover and Gaussian mutation) are used. In scattered crossover, the chromosome of the child is copied from the parent that is selected based on randomly created binary vector (1 selects one parent, whereas 0 selects the other parent). In Gaussian mutation, a random number generated with Gaussian distribution is added to the gene to generate the new individual. Two stopping criteria have been defined: (1) maximum number of iterations is set to 100 and (2) maximum number of consecutive iterations without any change in the objective function is set to 50.

3.2. Particle Swarm Optimisation (PSO)

PSO is another chromosome population-based search algorithm similar to GA. Unlike GA, PSO bases on the incorporation of knowledge found by the individuals to the search process of other individuals. Thus, individuals do not compete as in GA, but cooperate in PSO. The incorporation is performed by using the information from other individuals to define the direction and speed of the movement of other individuals in the search process. Thus, each individual defines its movement based on its own best position found and global best position found by the whole team.

Each individual stores its own best solution found so far during the search and moves towards the direction defined by its own and best of bests solutions found by others. The move is performed by adding the velocity vector to the current solution vector. The process is repeated until a predetermined minimum error is achieved or a predefined maximum number of iterations

has been exceeded. PSO has been used in many areas and more information can be found in Banks *et al* (2007).

In iteration, the move of a particle has been defined in different ways for binary or integer programming problems. Two methodologies using different solution and velocity update mechanisms for binary PSO have been compared (Camci, 2009b). One methodology uses binary operations such as and, or, or *x*-or for solution and velocity update, whereas the second one employs probability of change from 0 (1) to 1(0). The first methodology gives fast convergence, whereas the second one leads to better results with longer computational time. Thus, both methodologies have been serially performed in Camci (2009b) to have fast convergence to the better result. The solution and velocity updates have been performed with binary operations for the first 20 iterations as shown in (15) and (16).

$$V_{i,t+1} = or\Big(and\big(rand(1,m), xor\big(best_{global}, X_{i,t}\big)\big), and\big(rand(1,m), xor\big(best_i, X_{i,t}\big)\big)\Big) \quad (15)$$

$$X_{i,t+1} = xor\big(X_{i,t}, V_{i,t+1}\big) \quad (16)$$

where:

- r and $(1, m)$ is binary string randomly generated m bits;
- $\mathbf{V_{i,t}}$ is the velocity vector for individual i, at time t;
- $\mathbf{X_{i,t}}$ is the solution vector for individual i, at time t;
- $best_i$ is the best solution found by individual i;
- $best_{global}$ is the best of best solutions found by all individuals.

After the first 20 iterations, the solution and velocity updates have been performed based on the probability of change of a bit as shown in (17), (18), and (19).

$$V_{i,j}^1 = wV_{i,j}^1 + (-1)^{1-ibest} c_1 r_1 + (-1)^{1-gbest} c_2 r_2 \quad (17)$$

$$V_{i,j}^0 = wV_{i,j}^0 + (-1)^{ibest} c_1 r_1 + (-1)^{gbest} c_2 r_2 \quad (18)$$

$$x_{i,j}(t+1) = \begin{cases} \overline{x_{i,j}(t)} & \text{if } r_i < V_{i,j} \\ x_{i,j}(t) & \text{otherwise} \end{cases} \quad (19)$$

where

- $V_{i,j}^1$ is the velocity (change probability) of bit 1 at location j in individual i, at time t;
- $V_{i,j}^0$ is the velocity (change probability) of bit 0 at location j in individual i, at time t;
- *ibest* is the bit value (0 or 1) at location j for best solution of individual i;
- *gbest* is the bit value (0 or 1) at location j for best of bests solution of all individuals;
- c_1 and c_2 are constants;
- r_1 is a random number;
- $\overline{x} = 1$, if $x = 0$ or $\overline{x} = 0$, if $x = 1$.

3.3. Critical parameters in TMP

There are two critical parameters in TMP: Penalty for non-existing cities and number of visits.

Penalty for non-existing cities: GA operators (ie, crossover and mutation) may lead to city numbers that do not exist. For example, when a mutation is applied to the 13th character of the second chromosome in Table 1 (001 100 010 101 **0**11), the last city to be visited will be city 7, which does not exist; the chromosome becomes (001 100 010 101 **1**11). Similarly, if a crossover is applied to the chromosomes in Table 1 after the seventh character, the offspring (ie, 011 010 110 101 011 – 3 2 **6** 5 3, 001 100 000 001 101–1 4 **0** 1 5) will include the invalid cities 0 and 6. This result is likely when the number of cities is not a power of two. To prevent this result from happening, the value of the objective function is changed to a very big number when an invalid city is obtained in the solution.

Identification of the number of visits (K): In the TMP, the total number of visits may be different from the number of cities, which is different from TSP. As mentioned earlier, some cities may be visited more than once or may not be visited at all. The number of feasible solutions in TMP is then defined as n^K, where it is $n!$ in TSP. The computational complexity in TMP is much higher, if $K \geqslant n$. It is obvious that the parameter K is highly related with T, duration, in which the maintenance is scheduled. This new parameter (K) creates a new question of 'how to define the correct number of visits?' From a maintainer's point of view, the answer to this question is related to the time frame of the maintenance to be scheduled. The number of visits should be able to capture the maintenance plan duration. If a daily maintenance plan is performed for an 8-h work period, then the number of visits should be big enough to capture an 8-h period.

Since the parameters K and T are interrelated, one of them can be obtained using the other. Either of these parameters should be received as an input to the model. When T is received as an input, the calculation of K should be done within the model (ie, stops adding new city to visit when the total time exceeds T). This approach creates extra computation within the model. On the other hand, when K is given, T can be analysed after the solution is obtained. The cities scheduled for visit after the desired duration (T) can be removed from the schedule. K should be given big enough to capture the desired duration and small enough not to increase the computation unnecessarily. The disadvantage of this approach is to obtain different scheduled durations for different solutions. The size of the problem and the importance of the consistency in duration will define which approach to take. In this paper, the latter approach is used not to increase the computational burden within the model. Note that number of visits in practice may be less than K. When a city is visited consecutively, in practice only one city will be visited, which means the maintainer will be waiting in that city. For example, even if the schedule (1, 2, 2, 2, 1, 3) contains six visits, in practice only four visits (1, 2, 1, 3) have been obtained.

4. Results and discussion

This section includes two subsections: a case study on railway switch maintenance scheduling and a solution analysis of cases with different number of cities. The first subsection reports on a case study for maintenance scheduling of six railway switches. The second subsection discusses a detailed analysis of solutions for cases with cities from 4 to 32. Matlab (ver.7.10, optimisation toolbox ver. 5.0) is used as programming language on a computer with 2.66 GHz CPU and 3 GB RAM.

4.1. Case study on railway switches

Railway switches determine the direction of a train by moving rails. For management and maintenance purposes, railways are divided into responsibility areas, each of which includes a different number of switches. Switches in a responsibility area are located in various places, and their maintenance is managed by a distinct team. Maintenance scheduling of railway switches with prognostics information is a typical example of a TMP.

In this case study, one maintenance team is responsible for six switches (2 through 7 in Figure 4) and a starting point without a switch (1 in Figure 4). The connections between switches represent the railway. Maintenance operators can travel by rail only. As seen from the figure, some cities (switches) have direct rail connections, while some do not. The maintenance people are located in city '1', which does not have a railway switch to be maintained. They start and end at this location.

Table 2 gives the parameters used in this case study. Weibull distribution is used for failure probability estimations due to its capability to represent failure probability progression. The first five parameters are the Weibull distribution parameters, which are used to extract the forecasted failure probabilities. The first two parameters are for failure probability estimations after maintenance. These parameters are normally estimated using historical failure data. The next three parameters are used for forecasted failure probabilities starting from the current time, which will normally be received as input from prognostics modules. Shift parameters are used to shift the obtained probability curves in time to obtain the increased failure probability. The last four parameters are cost parameters.

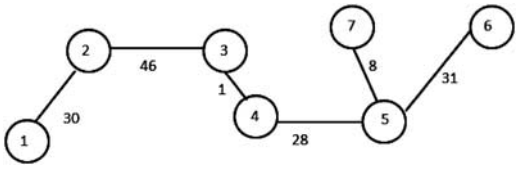


Figure 4 Railway switches.

This problem is solved using two scenarios. In the first scenario, maintenance is scheduled with the parameters given above. In the second scenario, the result of a detected failure in the TMP is illustrated by changing the failure probability of a switch. In each scenario, three groups of results will be reported: the global optimum for the TMP, the global optimum TSP, and the results of the TMP with the GA-based solution. To give the global optimum for the TMP, the TMP is restricted as a TSP in the sense that all of the cities will be visited exactly once. For example, TMP global for two cities is restricted to two solutions (ie, 1–2, 2–1), whereas TMP, in general, may have more solutions (eg, 1–1, 1–2, 2–1, 2–2). The total costs of all feasible solutions ($7! = 5040$ solutions) for the TMP global and TSP are calculated, and global optimum are reported. K value is defined as 7 in TMP.

Scenario 1: Table 3 displays the results obtained in this scenario. Two solutions give the global optimum in the TSP. TMP global is obtained by calculation of all permutations of visits with the restriction of visiting all the cities exactly once. GA-based TMP is obtained using the GA with the TMP formulation. The total travel time is 152 units, leading to a total cost of \$1520. K is chosen to be equal to the total number of cities ($K = 6$). In this way it can be checked if all cities will be scheduled for maintenance exactly once or some will be visited multiple times, while others will not be visited at all.

The global best solution in the TMP is presented as (2 3 4 5 7 6 1). Visit times column in the table have six sections separated by '/', each of which shows the visit time(s) of the cities in order. In GA-based TMP results, some cities have brackets in the visit times if a city is not visited at all or visited more than once. For example, the first solution of GA-based TMP indicates two visits to city 3 (at times 86 and 200) and no visit to city 6, which is represented as empty brackets. Last two columns give the value of the objective function for the best and worst solutions, respectively.

Table 2 Parameters used in case study 2

	2	3	4	5	6	7
Shape (after)	0.65	1.2	0.95	0.85	0.75	0.74
Scale (after)	710	960	1060	1080	1060	1484
Shape (before)	0.8	1.15	0.75	0.9	0.35	1
Scale (before)	137	83	79	94	127	112
Shift (before)	125	15	25	20	9	17
Fixed failure cost (\$)	15 250	14 850	14 700	15 050	14 759	15 406
Maintenance cost (\$)	151	149	148	153	149	149
Downtime cost (\$/time unit)	4.5	5.5	6.0	5.0	6.7	6.0
Unit distance cost to the node (\$/distance unit)	10	10	10	10	10	10

Table 3 Results of scenario 1

	Solution	*Visit times*	*Cost ($)*	*Best solution*	*Worst solution*
TSP	1 2 3 4 5 7 6	30/76/77/105/113/152	1520	1520	5730
	1 2 3 4 7 5 6	30/76/77/113/121/152	1520		
TMP global	2 3 4 5 7 6 1	30/86/97/135/202/153	80 056	80056	97320
GA-based TMP	2 3 4 5 7 3	30/[86 200]/97/135/[ ]/153	80 107	80107	80856
	2 3 4 5 7 3	30/[86 200]/97/135/[ ]/153	80 107		
	2 3 4 5 7 3	30/[86 200]/97/135/[ ]/153	80 107		
	2 3 4 5 7 3	30/[86 200]/97/135/[ ]/153	80 107		
	2 3 4 1 5 7	30/ 86/97/299/[ ]/317	80 268		
	3 4 5 7 3 2	246/[76 190]/87/125/[ ]/143	80 741		
	2 3 1 4 5 7	30/86/259/297/[ ]/315	80 268		
	2 3 4 5 7 3	30/[86200]/97/135/[ ]/153	80 107		
	2 3 1 4 5 7	30/86/259/297/[ ]/315	80 268		
	3 4 1 5 7 2	400/76/87/289/[ ]/307	80 856		

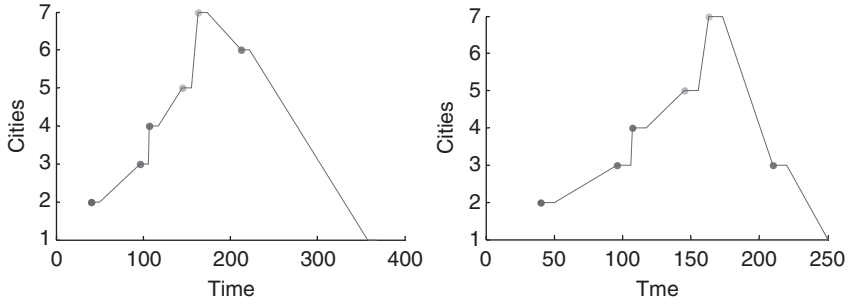


Figure 5 Route of the global best solution and GA solution in the TMP.

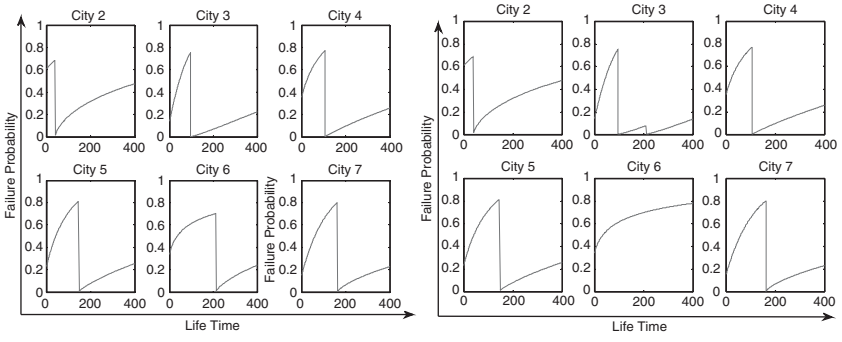


Figure 6 Failure probabilities with a maintenance schedule for global optimum solution and GA-based solution.

The GA was run 10 times with a population size of 150 and 150 generations, which is 50% more than the default values used in Matlab for mixed integer problems. The results are also reported in Table 3. The global optimal solution of the TMP for the given problem is (2 3 4 5 7 6 1), as displayed in the left of Figure 5. The *x*-axis gives the time, the *y*-axis gives the cities, and the line gives the travel routing. The best solution obtained from GAs gives the routing of cities '2 3 4 5 7 3', as displayed in the right of Figure 5. As seen from the figure, city 3 is visited twice, whereas city 6 is not visited. Figure 6 displays the failure probabilities of switches in cities with scheduled maintenance of global best and GA-found solution for effective comparison. The *x*-axis and the *y*-axis represent time and failure probabilities, respectively. The step in each city in the figure represents the maintenance time to be spent in each city. The maintenance time is assumed to be a constant for all cities in this paper. It is clearly shown that the failure probabilities are reduced with maintenance. Note that the failure probabilities before maintenance are likely to increase more rapidly than the failure probability after maintenance due to an incipient failure

detected. Prognostics results are more accurate and reliable when an incipient failure is detected.

The parameters used to evaluate the quality of the results are given in Table 4. Other parameters, except for Pglb and Pmn, are expected to be low.

The analysis of the GA results is given in Table 5. To evaluate the effect of the initial population, two types of runs are performed. In the first type (Run1_1 and Run1_3), the same initial population is used for all runs. The initial population is created randomly, and no expert knowledge is used. In the second type (Run1_2 and Run1_4), a different random initial population is used for each run. In Run1_1 and Run1_2, the size of the population and the maximum number of generations are defined as 150, whereas they are 750 in Run1_3 and Run1_4. Global optimum value is obtained by calculating all the permutations of visits assuming that every city will be visited exactly once. Note that this may not be the global optimum if we remove the one-visit limitation. However, the restricted global optimum will be a used as assessment criteria for the quality of the results found. As seen from the table, the GA could not find the global optimum in 10 runs with a population size of 150 and a maximum number of generations of 150($Pglb = 0$). The global optimum was found in 30 and 40% of the runs (3 and 4 out of 10 runs) when the size of the population was increased, with the price of approximately five times more computational time. With a population size of 150, the closeness of the GA's average results to the global best is 0.3% ($Cglb = 0.003$), and the closeness of the average to the found minimum by GA is 0.2% ($Cmn = 0.002$), which occurred in 50% of the runs ($Pmn = 0.50$). The closeness of the average to the maximum result found by the GA is 0.93% ($Cmx = 0.0093$), which occurred in 10% ($Pmx = 0.1$) of the runs. No significant effect of the various initial populations was observed. All computation times given in the paper are in seconds.

Table 4 Parameters used to evaluate the GA results

Parameters	*Description*	*Desired*
Ave	Average of costs obtained on all runs of the GA for the given dataset	Low
Fmin	Minimum cost obtained by the GA among all the runs for the given dataset	Low
Fmax	Maximum cost obtained by the GA among all the runs for the given dataset	Low
Cglb	Closeness of average to the global minimum	Low
Cmn	Closeness of average to the minimum found	Low
Cmx	Closeness of average to the maximum found	Low
Pglb	Percentage of runs in which the global min is obtained	High
Pmn	Percentage of runs in which the found min is obtained	High
Pmx	Percentage of runs in which the found max is obtained	Low
Ta,Tn,Tx	Average, minimum, and maximum computational time of all runs	Low

The population size and number of generations have increased together in Run1_1 through Run1_4. In order to see the individual effects of these parameters, Run1_5 and Run1_6 have been performed. In Run1_5, population size is 750 and number of generations is 150, whereas in Run1_6, number of generations is 750 and population size is 150. Computational time in Run1_5 has increased as in Run1_3 and Run1_4. This is not true in Run1_6. This means that the GA stops often due to the maximum number of iterations without any change in the objective function not due to maximum total number of iterations. In Run1_5, the global optimal is found with highest rate (60%) and closest approximation to the average (0.2%), whereas in Run1_6, it is 10%. This indicates increasing population size is more effective than increasing number of generations.

Scenario 2 (Increased expected failure consequence): As mentioned above, the TMP integrates the TSP with CBM, which requires continuous monitoring of assets. Expected failure consequence may increase due to increase in either failure probability (P_{π_i}) or in failure consequence (F_{π_i}, direct failure cost). Failure probability may increase due to a detected failure. Failure consequence may increase due to unavailability of a redundant system, which could replace the existing system in case of a failure. The second scenario demonstrates this concept using the same example as in the first scenario. Assume that expected failure consequence increases after 150 time units due to a scheduled maintenance for the redundant system. Any maintenance scheduled after this time will lead to increased failure cost. The TSP results are not reported because these results would not be different from those of the previous

Table 5 Analysis of GA results in scenario 1

	Gen	*Pop*	*Ave*	*Global* min			*Best performance*			*Worst performance*			*Comp time*		
				*G*min	*Pglb (%)*	*Cglb (%)*	*F*min	*Pmn (%)*	*Cmn (%)*	*F*max	*Pmx (%)*	*Cmx (%)*	*Ta*	*Tn*	*Tx*
Run1_1	150	150	80293	80 050	0	0.3	80 107	50	0.23	80 856	10	0.93	159	155	161
Run1_2	150	150	80 465	80 056	0	0.5	80 268	50	0.24	80 856	10	0.48	144	142	146
Run1_3	750	750	80 313	80 056	40	0.3	80 056	40	0.3	81 733	10	1.73	722	713	758
Run1_4	750	750	80 283	80 056	30	0.2	80 056	30	0.2	80 741	20	0.56	728	718	737
Run1_5	150	750	80 247	80 056	60	0.2	80 056	60	0.2	80 856	10	0.7	727	717	735
Run1_6	750	150	80 546	80 056	10	0.6	80 056	10	0.6	81 733	10	0.6	146	140	162

Table 6 Results of scenario 2

	Solution	*Visit times*	*Cost ($)*	*Best solution*	*Worst solution*
TMP global	3 6 5 7 4 2 1	308/76/251/187/146/205	85 723	85 723	1 557 091
GA-based TMP	1 3 1 1 6 7	[ ]/86/[ ]/[ ]/328/377	89360	85 723	89 360
	1 3 6 5 7 4	[ ]/86/261/197/156/215	86 170		
	3 6 5 7 4 2	308/76/251/187/146/205	85 723		
	3 6 5 7 4 2	308/76/251/187/146/205	85 723		
	3 6 5 1 7 4	[ ]/76/471/187/146/425	86 170		
	1 3 1 6 5 7	[ ]/86/[ ]/359/318/377	87 486		
	1 1 3 6 5 7	[ ]/96/[ ]/207/166/225	87 486		
	1 3 1 6 5 7	[ ]/86/[ ]/359/318/377	87 486		
	1 1 3 6 5 7	[ ]/96/[ ]/207/166/225	87 486		
	3 6 5 7 4 2	308/76/251/187/146/205	85 723		

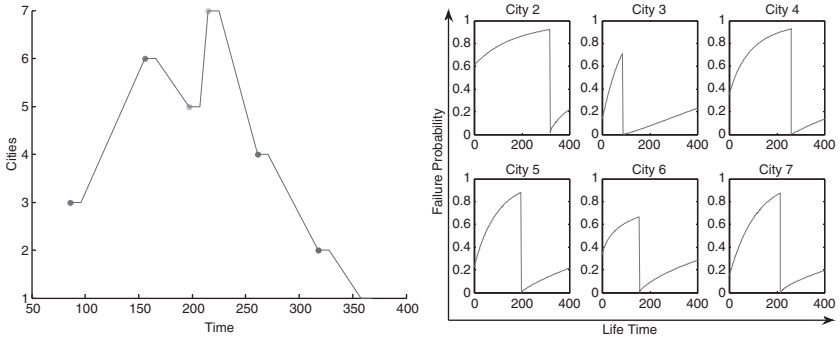


Figure 7 Maintenance schedule and failure probabilities in scenario 2.

Table 7 Analysis of the GA results in scenario 2

	Gen	*Pop*	*Ave*	*Global* min			*Best performance*			*Worst performance*			*Comp time*		
				*G*min	*Pglb (%)*	*Cglb (%)*	*F*min	*Pmn (%)*	*Cmn (%)*	*F*max	*Pmx (%)*	*Cmx (%)*	*Ta*	*Tn*	*Tx*
Run2_1	150	150	86 881	85 723	30	1.3	85 723	30	1.3	89 360	10	2.70	155	149	159
Run2_2	750	750	86 299	85723	30	0.6	85 723	30	0.67	87 486	20	1.30	776	761	788

scenario. The global best value is obtained by running all permutations of visits with the restriction of visiting all the cities exactly once. Table 6 displays the results of scenario 2. In this scenario, K is equal to 6 as in scenario 1.

As shown in Table 6 and Figure 7 (left), the global best solution is '3 6 5 7 4 2 1'. The maintenance of city 6 is scheduled at 146 time units by visiting city 6 as the second city instead of sixth. Figure 7 (right) shows the failure probabilities for the global best solution. It is clearly shown by this result that the TMP is able to use the forecasted failure probability information from CBM to produce an effective maintenance schedule. Table 6 also displays the results of the GA runs. As seen from the results, the GA is able to find the global optimum. Also, note that the results obtained from the GA may include multiple visits or no visits to some cities.

Table 7 displays the analysis of the GA results. The entry 'Run2_1' represents the GA runs with a population size of 150 and a maximum number of generations of 150, which are increased by a factor of 5 in 'Run2_2'. The GA is able to find the global best in 30% of the runs. The maximum cost obtained by the GA as the best solution is close to the global best by 2.7%, which occurred in 10% of the runs. The results of the GA can be enhanced (the closeness of the average best results to the global optimum can be reduced by 48%, ie, from 0.013 to 0.0067%, and the closeness of the worst result to the global optimum can be reduced by 52%, from 2.7 to 1.3%), at a price of a five-fold increase in computational time due to increasing the size of the population and the number of generations by a factor of 5.

4.2. Solution analysis with test data

This section reports the results of GA-based solutions for TMP using test data. The distances between cities are obtained from

Table 8 GA solution analysis for test data

	Average	*Best performance*			*Worst performance*			*Computational time*		
	Ave	*F*min	*Pmn (%)*	*Cmn (%)*	*F* max %	*Pmx (%)*	*Cmx (%)*	*Ta*	*Tn*	*Tx*
4 cities	39 760	39 387	50	0.94	41 321	10	3.78	24	24	25
5 cities	55 738	55 701	50	0.06	55 993	10	0.45	43	42	44
6 cities	66 557	66 189	10	0.55	67 147	10	0.88	102	100	105
7 cities	80 416	80 055	20	0.45	80 856	20	0.54	144	143	148
8 cities	94 408	94 262	40	0.15	94 948	10	0.57	196	193	199
9 cities	101 314	100 968	20	0.34	101 825	20	0.50	253	249	257
10 cities	103 059	103 019	20	0.03	103 296	10	0.23	435	427	441
11 cities	106 018	105 815	10	0.19	106 378	20	0.34	536	525	550
12 cities	113 602	113 307	10	0.26	114 360	20	0.66	622	632	613
13 cities	119 529	118 409	10	0.94	120 484	10	0.79	732	740	723
14 cities	125 481	124 200	20	1.03	126 265	10	0.62	867	858	875
15 cities	129 785	128 902	10	0.68	130 859	20	0.82	1022	1008	1044
16 cities	134 409	133 488	10	0.69	134 854	60	0.33	1177	1161	1198
17 cities	140 878	139 773	10	0.79	141 562	40	0.48	1343	1324	1369
18 cities	147 133	146 210	10	0.63	147 482	10	0.24	1771	1736	1809
19 cities	150 723	149 465	10	0.84	151 307	10	0.39	1985	1899	2118
20 cities	154 427	153 418	10	0.65	154 914	10	0.31	2192	2134	2243
21 cities	158 100	157 017	10	0.68	160 018	10	1.20	2451	2379	2499
22 cities	161 454	160 180	10	0.79	161 914	10	0.28	2699	2662	2733
23 cities	166 611	165 712	10	0.54	167 143	10	0.32	2955	2784	3004
24 cities	172 243	171 056	10	0.69	172 497	10	0.15	3318	3189	3496
25 cities	176 859	175 849	10	0.57	177 565	10	0.40	3609	3546	3660
26 cities	181 811	180 693	10	0.61	182 465	10	0.36	3916	3770	3972
27 cities	185 306	184 210	10	0.59	185 845	10	0.29	4235	4184	4311
28 cities	189 411	188 333	10	0.57	189 716	20	0.16	4613	4539	4650
29 cities	193 735	192 150	10	0.82	194 403	10	0.34	4932	4845	5001
30 cities	195 426	194 097	10	0.68	196 391	10	0.49	5319	5253	5433
31 cities	199 473	198 730	10	0.37	2002 79	10	0.40	5731	5674	5872
32 cities	201 990	201 058	10	0.46	202 922	10	0.46	6235	6112	6327

OR library (Beasley, 1990). The other parameters such as costs, Weibull distribution parameters are randomly generated. Basic insight rather than an expert evaluation such as failure cost being 10 times or more greater than maintenance cost with some randomness is used for generating the data. The first test data include only four cities. The number of cities in the test data is increased from 4 to 32 (first 32 cities in the data set are used) by adding the next city in the data set to the previous test data. The presented GA-based method is run 10 times for each number of cities. The number of generations and population size are defined as $n \times log_2^n \times 50$, where n is the number of cities, with the idea of growing the population and number of generations more than the product of some constant and n.

The parameter K is predefined as equal to n for these runs and T is left as it is obtained (solutions may have different T). The total number of feasible solutions is n^n. Because the global optimal solutions for the datasets are not known, the quality of results could not be evaluated based on the approximation to the global optimal solution. Table 8 displays some quality evaluation parameters to illustrate a baseline for the solution of TMP that can be used as a benchmark for future studies. Found minimum (maximum) is the smallest (greatest) value of the objective function among 10 runs of GA. Fmin and Fmax are the actual values of minimum and maximum found.

Figure 8 displays the change of computational time and average optimum values found with the increase of number of cities. x-axis displays the different number of cities and y-axis represents the computational time and value of the objective function obtained. The actual values are not displayed since they are already reported in the tables and the goal of this figure is to visualise the changes with increase in the given parameter. As seen from the figure, the computational time increases exponentially, whereas average of optima found follows a logarithmic function. This indicates that GA stops more due to maximum number of total iterations than due to maximum number of iterations allowed without any change in the objective function as the size of the problem increases. The shift of both of computational time and average of optima found to linearity indicates an improvement for future studies.

As seen from Table 8, the minimum value could be found multiple times for problems with less than 10 cities (Pmn > %10). However, this is not true for bigger problems (Pmn = % 10) (except with 14 cities). Future studies may be interested in increasing the repeatability of minimum found for higher

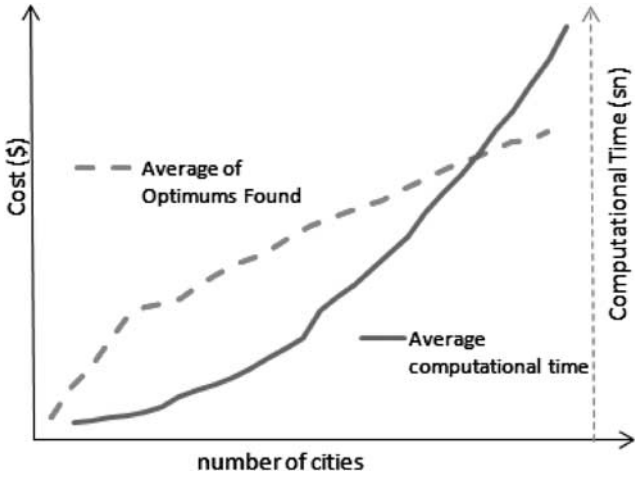


Figure 8 Change of computational time and optimum found by increasing number of cities.

Table 9 Results of GA for different K values

	Average	*Best performance*		*Worst performance*		*Computational time*		
K	*Ave*	*F*min	*Pmn (%)*	*F*max	*Pmx (%)*	*Ta*	*Tn*	*Tx*
1	91 641	91 641	100	91 641	100	24	24	24
2	87 823	87 792	90	88 098	10	55	55	55
3	84 714	84 670	90	85 100	10	93	92	94
4	82 022	81 733	70	83 003	10	137	136	141
5	80 738	80 268	50	81 733	20	184	181	186
6	80 320	80 056	20	80 741	20	242	234	237
7	80 232	79 835	20	80 856	10	295	290	303
8	80 154	79 835	20	80 741	10	361	352	407
9	80 022	79 835	10	80 107	10	423	419	432
10	80 069	79 835	50	80 741	20	499	486	507
11	80 009	79 835	10	80 268	10	569	559	581
12	79 879	79 835	40	80 056	10	654	644	662
13	79 855	79 835	60	79 884	40	743	735	750
14	79 894	79 835	50	80 056	20	830	813	846
15	79 899	79 835	60	80 107	20	934	910	954

number of cities. The maximum value found was not repeated for the problems with 4, 5, 6, and greater than 17 cities (Pmx = 10). However, it is repeated for problems with city size between seven and 17. When the problem size increases the worst result found was repeated, which is bad in the sense that we have less better results and good in the sense that the results obtained are consistent. When the size of the problem is increased more ($n > 17$), the probability of obtaining the same result reduces, which causes inconsistency.

Parameters Cmn and Cmx in Table 8 indicate the closeness of average of results to the minimum and maximum found, respectively. When the problem size is low, the minimum is close to the average indicating that more results are close to minimum. When the problem size is more than 10, the maximum found is closer to the average indicating that more results are close to the worst found. As seen from Table 8, the average gets closer to the maximum, when $n > 10$. The cut-off points (ie, $n > 10$ for Cmn and CMx and $n > 17$ for Pmn and Pmx) can be used for comparison of new methods. It is expected that better methods should push the cut-off points forward.

The number of visits (K) is defined as the number of cities (n) in previous examples of TMP. K is analysed by running GA for K from 1 to 15 *for* $n = 7$. Ten runs have been performed for each K value and results are presented in Table 9. When $K = 1$, only one maintenance will be performed and the city whose failure costs the most will be selected for maintenance. As seen from the Table 9, the result is consistent and the same city is selected for maintenance in all runs when $K = 1$. It is also observed from the table that the total cost declines when K is increased, since the failure cost is reduced by performing maintenance in more cities. The total cost declines until $K = 7$, than stays constant when $K > 7$, which surprisingly equals to the number of cities. When the total cost declines with increased K, the rate of finding the minimum (*Pmn*) decreases. When K becomes

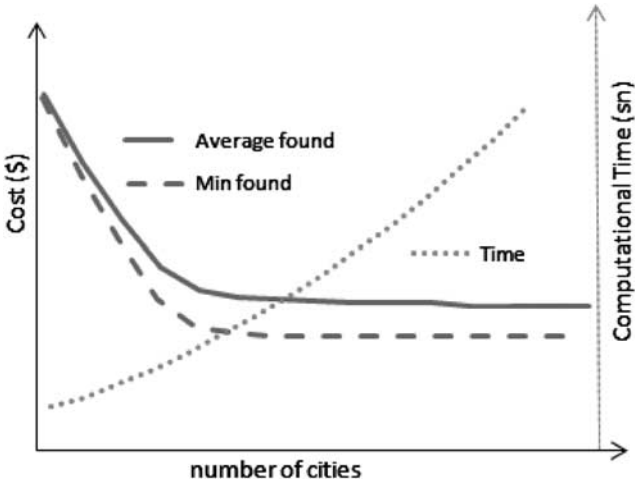


Figure 9 Change of average computational time, average and minimum of found results with K.

greater than 7, the total cost stays constant and *Pmn* increases with increasing computational time. We can conclude that if an expert recommendation exists about the estimation of K, then a larger number than the recommendation should be used for K, since TMP may find the optimal K with the cost of increased computational time. In other words, TMP cannot recommend a solution with more than K visits; however, it can recommend a solution with less than K visits. Remember the comment on visiting the same city consecutively means visiting that city only once in practice.

Figure 9 displays the changes in average of computational time, minimum, and average of found results. As seen from the figure, the computational time consistently increases with the increase in K. The minimum found reduces rapidly and stays constant when K reaches to optimal value. However, K might be increased more to increase the probability of obtaining the global optimal.

Even though optimum K is equal to number of cities in the given example, it is difficult to make a general statement about K. When Mean Time To Failure (MTTF) and the cost of failure of all cities are close, then it is expected that all cities will be visited once. However, when MTTF of one city is half of the others (with close failure costs), then the city may be visited twice, which may make K equal to $n + 1$. Similarly, if the failure cost of a city is much more than the failure costs of other cities given close MTTF, the city may be visited more than the others. If the maintenance cost of a city is close to its failure cost, then the city may not be visited at all leading it to failure, if no availability constraint is introduced.

4.3. Comparison of PSO and GA

In this subsection, the results of TMP with 8, 16 and 32 cities solved by GA and PSO will be compared. Note that the purpose of this subsection is not to fully compare and analyse the advantage and disadvantages of GA and PSO, but to give insights of using GA or PSO for the defined TMP. The parameters used in PSO are given in Table 10. The other parameters required for TMP are obtained using OR library or randomly generated numbers, as mentioned before.

Average, minimum, and maximum costs and computational times are reported in Table 11. As seen from the table, GA gives better results compared to PSO. However, the convergence time is much longer. This does not necessarily mean that increasing PSO may converge to a better solution when more time is allowed. The iterations have mostly ended in PSO due to the insufficient change in the objective function.

Even though there are many parameters to analyse in comparison of PSO and GA and the focus of this paper is not to fully compare them, the findings in their comparison support similar studies such as Camci (2009b). Binary PSO is not yet mature enough to compete with GA and needs further developments.

5. Conclusions and future work

This paper presents a new problem: the TMP. The TMP takes the latency to each visited city as an input to a function and aims to minimise the sum of these functions. The function in the TMP uses the latency to calculate the expected cost of maintenance, failure, downtime, and travel. This paper also presents a GA and PSO-based solution to the TMP and demonstrates the results of a study and results on test datasets.

Table 10 Parameters used

Number of cities	*Maximum iterations*	*Size of population*	*Time period (T)*	*Total runs*
8	100	75	250	10
16	200	150	500	10
32	400	300	1000	5

Table 11 Results of GA and PSO

	PSO		*GA*	
	Total cost	*Time*	*Total cost*	*Time*
Number of cities: 8				
Average	89385	32.61	80154	361
Max	91084	42.55	80741	407
Min	88393	24.09	79835	352
Number of cities: 16				
Average	142090	274	134409	1177
Max	143241	541	134854	1198
Min	141231	155	133488	1161
Number of cities: 32				
Average	287960	2228	201990	6235
Max	289617	2695	202922	6327
Min	285415	1763	201058	6112

Although the presented PSO and GA-based solutions give good results, the TMP is open to new enhancements and heuristic methods. The new solutions should be able to incorporate problems with more cities and to model practical issues and constraints related to real-time situations in maintenance planning such as daily work periods, stochastic maintenance, and travel times. For example, maintenance schedule should be tied with the daily work duration. More complex real situation may include different vehicle alternatives. For example a maintainer responsible for the maintenance of switches in a responsibility area uses a small rail-car. He may also use roads to travel when needed. When he travels to a city with this rail-car and goes to a different city by road (not railway), he needs to come back to the city where he left the rail-car. Besides, the maintenance operator needs to visit the starting point at some point to take parts and materials to be used during maintenance and repair. He can take a limited amount of these goods when he visits the starting point. Thus, the model needs to limit the time or the number of visits to the other cities without visiting the starting point. These types of constraints should be studied further.

Acknowledgements—This research was partially supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under project 108M275. In 2009, the problem discussed in this paper has been recommended as MSc thesis to Musa Karakas under supervision of the author and co-supervision of Mehmet Sevkli. After initial study, Musa Karakas has decided not to work on this topic. The author acknowledges their initial contribution in developing the idea.

References

Angelelli E, Mansini E and Vindigni M (2011). Look-ahead heuristics for the dynamic traveling purchaser problem. *Computers & Operations Research* **38**(12): 1867–1876.

Balaprakash P, Birattari M, Stützle T and Dorigo M (2010). Estimation-based metaheuristics for the probabilistic traveling salesman problem. *Computers & Operations Research* **37**(11): 1939–1951.

Balas E (1995). The prize collecting traveling salesman problem: II, Polyhedral results. *Networks* **25**(4): 199–216.

Banks A, Vincent J and Anyakoha C (2007). A review of particle swarm optimization. Part I: Background and development. *Natural Computing* **6**(4): 467–484.

Beasley JE (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society* **41**(11): 1069–1072.

Bianco L, Mingozzi A and Ricciardelli S (1993). The traveling salesman problem with cumulative costs. *Networks* **23**(2): 81–91.

Blum A, Chalasani P, Coppersmith D, Pulleyblank B, Raghavan P and Sudan M (1994). The minimum latency problem. *Proceedings of the twenty-sixth annual ACM symposium on Theory of Computing—STOC* '94, Montreal, Canada, pp 163–171.

Bontoux B, Artigues C and Feillet D (2010). A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. *Computers & Operations Research* **37**(11): 1844–1852.

Camci F (2009a). System maintenance scheduling with prognostics information using genetic algorithm. *IEEE Transactions on Reliability* **58**(3): 539–552.

Camci F (2009b). Comparison of genetic and binary particle swarm optimization algorithms on system maintenance scheduling using prognostics information. *Engineering Optimization* **41**(2): 119–136.

Camci F and Chinnam RB (2006). Hierarchical HMMs for autonomous diagnostics and prognostics. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Vancouver, Canada, pp 2445–2452.

Camci F and Chinnam R (2010). Health-state estimation and prognostics in machining processes. *IEEE Transactions on Automation Science and Engineering* **7**(3): 581–597.

Carr MJ and Wang W (2011). An approximate algorithm for prognostic modelling using condition monitoring information. *European Journal of Operational Research* **211**(1): 90–96.

Casazza M, Ceselli A and Nunkesser M (2012). Efficient algorithms for the double traveling salesman problem with multiple stacks. *Computers & Operations Research* **39**(5): 1044–1053.

Chatterjee S, Carrera C and Lynch L (1996). Genetic algorithms and traveling salesman problems. *European Journal of Operational Research* **93**(3): 490–510.

Das TK and Wortman MA (1993). Analysis of asymmetric patrolling repairman systems. *European Journal of Operational Research* **64**(1): 45–60.

Dell'Amico M, Maffioli F and Värbrand P (1995). On prize collecting tours and the asymmetric travelling salesman problem. *International Transactions on Operations Research* **2**(3): 297–308.

Dohn A, Rasmussen MS and Larsen J (2011). The vehicle routing problem with time windows and temporal dependencies. *Networks* **58**(4): 273–289.

Erdoğan G, Cordeau J-F and Laporte G (2007). The attractive traveling salesman problem. *European Journal of Operational Research* **203**(1): 59–69.

Erdoğan G, Battarra M, Laporte G and Vigo D (2012). Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs. *Computers & Operations Research* **39**(5): 1074–1086.

Fakcharoenphol J, Harrelson C and Rao S (2007). The k-traveling repairman problem. In *ACM Transactions on Algorithms (TALG)* **3**(4): Article 40.

Feillet D, Dejax P and Gendreau M (2005). Traveling salesman problems with profits. *Transportation Science* **39**(2): 188–205.

Fischetti M, Laporte G and Martello S (1993). The delivery man problem and cumulative matroids. *Operations Research* **41**(6): 1055–1064.

Glomvik Rakke J, Christiansen M, Fagerholt K and Laporte G (2012). The traveling salesman problem with draft limits. *Computers & Operations Research* **39**(9): 2161–2167.

Gouveia L, Paias A and Voß S (2011). Models for a traveling purchaser problem with additional side-constraints. *Computers & Operations Research* **38**(2): 550–558.

Jardine AKS, Lin D and Banjevic D (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing* **20**(7): 1483–1510.

Jothi R and Raghavachari B (2007). Approximating the k-traveling repairman problem with repairtimes. *Journal of Discrete Algorithms* **5**(2): 293–303.

Laporte G and Palekar U (2002). Some applications of the clustered travelling salesman problem. *The Journal of the Operational Research Society* **53**(9): 972–976.

Moon C, Kim J, Choi G and Seo Y (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research* **140**(3): 606–617.

Nagarajan V and Ravi R (2012). Approximation algorithms for distance constrained vehicle routing problems. *Networks* **59**(2): 209–214.

Potvin J-Y (1996). Genetic algorithms for the travelling salesman problem. *Annals of Operations Research* **63**(3): 339–370.

Punnen AP (2002). The traveling salesman problem: Applications, formulations and variations. In: Gutin G and Punnen AP (eds). *The Traveling Salesman Problem and its Variations*. Kluwer Academic Publishers: Dordrecht, The Netherlands, pp 1–28.

Wen M, Krapper E, Larsen J and Stidsen TK (2011). A multilevel variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem. *Networks* **58**(4): 311–322.

Xiao Y, Zhao Q, Kaku I and Xu Y (2012). Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research* **39**(7): 1419–1431.

Yu W and Liu Z (2011). Single-vehicle scheduling problems with release and service times on a line. *Networks* **57**(2): 128–134.

Vansteenwegen P, Souffriau W, Vanden Berghe G and Van Oudheusden D (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research* **36**(12): 3281–3290.

Received 11 February 2013;
accepted 13 June 2013 after two revisions