A New Hybrid Parallel Simulated Annealing Algorithm for Travelling Salesman Problem with Multiple Transporters

Parham Azimi[a,*], Ramtin Rooeinfar[b], Hani Pourvaziri[b]

[a] *Assistant Professor, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
[b] *Msc, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
Received 8 September, 2013; Revised 20 November, 2013; Accepted 13 March, 2014

Abstract

In today's competitive transportation systems, passengers search to find traveling agencies that are able to serve them efficiently considering both traveling time and transportation costs. In this paper, we present a new model for the traveling salesman problem with multiple transporters (TSPMT). In the proposed model, which is more applicable than the traditional versions, each city has different transporting vehicles and the cost of travel through each city is dependent on the transporting vehicles type. The aim is to determine an optimal sequence of visited cities with minimum traveling times by available transporting vehicles within a limited budget. First, the mathematical model of TSPMT is presented. Next, since the problem is NP-hard, a new hybrid parallel simulated annealing algorithm with a new coding scheme is proposed. To analyze the performance of the proposed algorithm, 50 numerical examples with different budget types are examined and solved using the algorithm. The computational results of these comparisons show that the algorithm is an excellent approach in speed and solution quality.
Keywords: Traveling salesman problem; Transporter vehicles; Budget constraint; Mathematical programming;Simulated annealing algorithm.

1. Introduction

The travelling salesman problem (TSP) is one of the most important problems in combinatorial optimization. The inputs are a collection of cities and the travel cost between each pair of cities. The purpose of the TSP is to find out the cheapest way of visiting all the cities only once and returning to the first city. Practical applications of the TSP consist of many problems in science, technology and engineering, such as vehicle routing, wiring, scheduling operations, flexible manufacturing, VLSI layout and etc (Lawler et al., 1985; Rego et al., 2011).

The travelling salesman problem with multiple transporters (TSPMT) is a new type of the classical TSP in which the salesman must visit all cities considering the available transporting vehicles in each city using given transportation cost. The TSPMT involves determining a tour starting and ending at the depot, and visiting each node exactly once. The total cost of traveling by each vehicle (such as: train, bus, airplane, and etc.) should not be greater than the total available budget. The objective of this problem is to minimize the total travelling time assuming limited budget.

The TSPMT is more intractable than the classical TSP versions; so, it is a NP-hard problem. Therefore, there is no exact algorithm capable of solving all instances of the problem in a reasonable time, especially in large-sized problems. There are various approaches and numerous approximate methods which have been developed so far to solve TSP with various conditions. Many researches have been presented under both exact and heuristic methods to solve TSP. Exact methods include cutting plane, liner programming (LP) relaxation techniques (Dantzig et al., 1954), branch and bound algorithm (B&B) (Padberg and Rinaldi, 1980), B&B based on assignment problem relaxation (Held and Karp, 1971; Balas and Christofids, 1981), and dynamic programming (Crowder and Padberg, 1980; Grötschel and Holland, 1991; Ergan and Orlin, 2006). However, small-sized problems can be solved by exact methods. On the other hand, large-sized problems have been solved using heuristic and probabilistic method such as 2-opt, 3-opt (Lin and Kernighan, 1973), Markov chain(Martin et al., 1991) and

* Corresponding author E-mail: p.azimi@yahoo.com

metaheuristic algorithms such as Tabu search (Knox, 1989; Glover, 1990; Pedro et al., 2013), genetic algorithm (Goldberg and Lingle, 1985; Grefenstette et al., 1985; Hopfield and Tank, 1985; Goldberg, 1989; Jog et al., 1989; Whitley and Starkweather, 1989; Braun, 1991; Xing et al., 2008;Kuroda et al., 2010; Albayrak and Allahverdi, 2011; Majumdar and Bhunia, 2011; Nagata and Soler, 2012) and ,neural networks (Zhang et al., 2012), particle swam optimization, simulated annealing algorithm (Kirkpatrick et al., 1983; Bonomi and Lutton, 1984; Kirkpatrick and Toulouse, 1985; Lam, 1988; Marinakis and Marinaki, 2010) , ant colony optimization (Branke and Guntsch, 2004; Bianchi, 2006; Bontoux and Feillet, 2008; Jun-man and Yi, 2012; Mavrovouniotis and Yang, 2013), memetic algorithm (Bontoux et al., 2010), scatter search (Liu, 2008) and etc.To the best of our knowledge the TSPMT has never been previously studied. Our main contribution is to present the TSPMT model and solve it using a new hybrid parallel simulated annealing algorithm. The rest of the paper is organized as follows: In section 2, the mathematical model of the classical TSP and TSPMT are presented. In section 3, the solution methodologies such as direct parallel simulated annealing (PSA1) and parallel simulated annealing algorithm (PSA2) are specially explained. In section 4, the computational results have been presented for small, medium, and large sized problems in which we have compared the results of PSA1with PSA2 for 50 numerical examples with three different budget types. Also, the computational results of PSA1 and PSA2 are compared to LINGO software results for small-sized problems. Finally, conclusions and future researches are presented in section 5.

2. Mathematical Formulations

There are a wide range of formulations of TSP in the literature with different assumptions, constraints and properties. In this section, we present two model formulations, the first model is a classical form of the symmetric TSP and the second one is about TSPMT model which has been developed in this research.

2.1. The classical TSP problem

In the classical symmetric TSP, the distance/cost between each pair of cities is the same in each direction. The classical closed tour TSP can be formulated as follows (Rego et al., 2011):

Model 1:

Minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall\, j \in N \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\, i \in N \qquad (3)$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in N \qquad (4)$$

In this formulation, the objective function is to minimize the total traveling costs all over the arcs used to complete the tour. Constraints (2) and (3) are the standard assignment constraints. In addition, subtours elimination constraints (*SECs*) are needed.

2.2. *Sub tours eliminating constraints (SECs)*

A key part of a TSP is to make sure that the tour is continuous. Without such constraints we often will get solutions containing degenerate tours between intermediate nodes and not connected to the starting city. The originally *SECs* were firstly presented by Dantzing et al. (1954):

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - 1 \qquad \forall S \subseteq N/\{1\}, S \neq \emptyset \qquad (5)$$

Unfortunately this formulation creates an exponential number of constraints and becomes impractical even for small problems. A different *SCE* proposed by Miller et al.(1960) which has only a maximum of $(n\text{-}2)^2$ constraints, at the disadvantage of a weak LP relaxation:

$$u_i - u_j + 1 \le (n-1)(1-x_{ij}) \quad \forall (i,j) \in A, : i,j \neq 1 \qquad (6)$$

In (6) a new set of variables $U=\{u_i : i \in N,\ i \neq 1\}$ is required. The u_i are arbitrary real numbers, but it can be ranked to non-negative integers representing the sequence in which the nodes are being visited.

2.3. The TSP problem with multiple transporters (TSPMT)

The TSP with multiple transporters (TSPMT) is an extension of the classical TSP in which a salesman can visit all cities by different transporting vehicles with given cost. Each type of transporting vehicles has special cost and traveling time. The purpose is to find the minimum total traveling time with least budget consumed. The key assumptions of the TSPMT are:

- Node 1 is required to be the basic observing city;
- They make sure that every city visited belongs to a tour connected to the base city, thereby eliminating subtours;
- All nodes should be visited only once;
- Different transporting vehicles are assumed to travel between nodes;
- Transportation costs between nodes are dependent on vehicles type and are known;
- Total budget to travel between each node by each vehicle is known.

From the graph theory point of view, the TSPMTcan be defined on a graph $G=(V, A)$, where $V=\{v_1,\ldots, v_n\}$ is a set of n vertices (nodes) and $A=\{(v_i , v_j)_r \mid v_i, v_j \in V , i \neq j \}$ is a set of arcs, together with a non-negative time matrix $T = (t_{ij}^r)$ which means that the traveling time from node I to node j by transporting vehicle r, and non-negative cost matrix $W = (w_{ij}^r)$ that indicates the traveling cost from node i to node j by transporter vehicle r is associated with A. The common definition of the set of decision variables is $X=\{x_{ij}^r: i, j, r \in N, i \neq j \}$, where $x_{ij}^r= 1$ if the salesman travels from node i to node$j$ by transporting vehicle (node i is visited immediately before node j), and 0 otherwise. Our problem is considered to be symmetric TSP (STSP) which $t_{ij}^r=t_{ji}^r$ for all $(v_i,v_j)_r \in A$. Elements of A are often called edges (rather than arcs) in this case. The STSP consists in determining the Hamiltonian cycle (circuit), often simply called a tour of minimum cost. In this model, index w indicates the traveling cost, and index Q indicates the total available budget. The objective function and constraints of TSPMT are presented below:

Model 2:

Minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{n} x_{ij}^r\, t_{ij}^r \qquad (7)$$

$$\sum_{i=1}^{n}\sum_{r=1}^{n} x_{ij}^r = 1 \qquad \forall j \in N \qquad (8)$$

$$\sum_{j=1}^{n}\sum_{r=1}^{n} x_{ij}^r = 1 \qquad \forall i \in N \qquad (9)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{n} x_{ij}^r w_{ij}^r \leq Q \qquad (10)$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}^r)\ \ \forall (i,j,r,k) \in A, : i,j = 1 \qquad (11)$$

$$x_{ij}^r \in \{0,1\} \quad \forall i,j,r,k \in N \qquad (12)$$

The objective function of this model (7) is to minimize total transporting time along all the arcs used to complete the tour with different vehicles. Constraints 8 and 9 are the standard assignment constraints assuming different transporting vehicles. Constraint 10 ensures that total transporting cost should not be greater that total available budget. Constraint 11 prevents to make subtours in model. As we mentioned before, since the classical TSP (mode1) belongs to the class of NP-hard problems (Lawler et al., 1985), the TSPMT (model 2)is also such an NP-hard one. Therefore, the large-sized problems cannot be solved with exact algorithms in a reasonable time, so we should use heuristic or metaheuristic algorithms.

3. Solution Algorithms

In this section, we provide a novel solution algorithm for TSPMT problem. To this aim, the hybrid parallel simulated annealing with a new coding scheme is developed. First, we present simulated annealing algorithm, then parallel version of our simulated annealing algorithm such as direct parallel simulated annealing algorithm (PSA1) and hybrid parallel simulated annealing algorithm (PSA2) are specially explained.

3.1. Simulated annealing algorithm in general

Simulated Annealing (SA) is a stochastic optimization method that is based on iterative strategy and surely one of the first algorithms that has an explicit strategy to avoid local minimal. The origin of the algorithm is in statistical mechanism (Metropolis algorithm) and it was first presented by Kirkpatrick et al. (1985). The fundamental idea is to allow moves to be resulted in the worse solution than the current solution (uphill moves) in order to escape from local minimal solution. The probability of doing such a move is decreased during the search process. The algorithm is started by generating an initial solution and by initializing the so-called temperature parameter T. Then, the following is repeated until the termination condition is satisfied:

A solution j' from the neighborhoods of the current solution ($N(j)$)is randomly sampled and it is accepted as new current solution based on $f(j)$, $f(j')$ and T. j' replaces j if$f(j') < f(j)$ or, in case $f(j') \geq f(j)$, with a probability which is a function of T and $f(j') - f(j)$. The probability is generally computed by the Boltzmann function: $exp(-(f(j') - f(j))/T)$. The temperature T is decreased during the search process, thus, at the beginning of the search, the probability of accepting uphill moves is high and then it gradually decreases to converge to near optimum solution. This means that the algorithm is capable to utilize two main strategies: random walk and iterative improvement. In the first phase of the search, the move toward improvements is low and it permits the exploration of the search space; this erratic component is slowly decreased. Therefore, it results in the search to converge to a local minimum. The probability of accepting uphill moves is controlled by two factors: the difference of the objective functions and the temperature. In other words, at fixed temperature, a higher difference $f(j')- f(j)$, leads to decrease the accepting probability of a move from j to j'.

The SA algorithm is divided in four basic components as below:

a) Initial configuration: a model of what a legal placement (configuration) is. This represents the possible problem solutions over which we will search for a good answer.

b) Move generation function: a set of allowable moves to reach all feasible configurations. These moves are the computations that must perform to move from

configuration to configuration during the annealing process.
c) Cost function: to measure how well any given placement configuration is.
d) Cooling schedule: to anneal the problem from a random solution to a good, frozen, placement. This process is started with hot temperature and rules to determine when the current temperature should be decreased, by how much the temperature should be lowered and when annealing should be terminated.
e) Stopping criterion: to when the algorithm must be stopped and define when the current temperature reaches to predefined temperature.

3.2. Hybrid Parallel simulated annealing algorithms for TSPMT

The SA performance convergence is usually affected by the quality of the initial solution. Sometimes SA algorithm is combined with a new coding scheme of other metaheuristics in order to reduce some deficiencies to find near optimum solution in a reasonable time. Because of the variety of search space of each SA metaheuristics, every SA algorithm searches in a different ways. This approach can be implemented on parallel processors in a completely asynchronous way. The essential components of our direct parallel simulated annealing algorithm (PSA1) and hybrid parallel simulated annealing (PSA2) are similar except in generating initial solutions as described below (Onbasoglu et al., 2001):

3.2.1. Initial solution of parallel simulated annealing algorithm (PSA1)

The initial solution is randomly generated by allocating a value to each dimension i, $i = 1, ...,N$, in interval $[L_b, U_b]$, where L_b and U_b are the lower and upper bounds for variable i, respectively.

3.2.2. Initial solution of hybrid parallel simulated annealing algorithm (PSA2)

First, we solve the problem using mathematical programming method to find primal inputs of initial solutions for SA algorithm. Since we cannot solve the general TSPMT problem, we consider TSPMT with relaxed conditions situation. In the literature, there are two possible ways to relax: to allow each vertex to visits many times and to relax the binary constraints. We consider the second way to change the model to a linear one which relaxes the binary constraints. Next, the relaxed model was solved by LINGO software. Then, we used the optimum solutions of linear model as a probability distribution. As the results show, we find good tours without considering budget constraint. Therefore, the initial solutions found in this way would not satisfy the budget constraint.

To make the solutions feasible, a heuristic algorithm has been developed as follows:
Step 1) Define $K \subset N$ as a subset of vectors which participate in tour and set K={current solution}. Denote C as the set of correspondent transportation cost of the elements in K.
Step 2) Define $T \subset K$ as a subset of vectors which move by the expensive vehicle type.
Step 3) From the T set select a path randomly. Assume that vector is moved by a cheaper vehicle type and delete the relevant vector from T.
Step 4) Consider a cheaper vehicle type for the first element in T. Then, calculate diminution in usable upon new fitness function and defined it as discount rate.
Step 5) Calculate the traveling cost of a new vector. If the whole traveling cost satisfies the budget constraint, stop, otherwise go to step 3.
The other components of our direct parallel simulated annealing algorithm (PSA1) and hybrid parallel simulated annealing algorithm (PSA2) are the same and described as follows:

3.2.3. Neighborhood generation

The current solution x_j are changing to the neighbourhood solution x_{j+1} in iteration j by selecting a random dimension i^* and calculating the variable values of the neighbour by the following equations:

If $i=i^*$

$$x_i^{j+1} = x_i^j + \left(yb_i - x_i^j\right), \quad sgn(y - 0.5) < 0, \qquad (13)$$

Or

$$x_i^{j+1} = x_i^j + \left(x_i^j - lb_i\right), \quad sgn(y - 0.5) \geq 0, \qquad (14)$$

Else

$$x_i^{j+1} = x_i^j, \qquad where\ y \in U[0,1] \qquad (15)$$

Where y is a random variable uniformly distributed between [0,1]. In this approach, variable i^* is randomly selected and a decision is made conventionally if the value of the variable in the selected dimension is to be decreased or increased. Then, a random amount is added or subtracted from the current variable value without violating the corresponding upper and lower bound while the remaining values of the variable remain constant.

3.2.4. Cooling rate

We reduce the temperature using a factor of 0.95. We found in the preliminary tests that this cooling rate leads into good solutions by tuning this initial value which is presented in subsection 4.1.

3.2.5. Acceptance of neighbor solution

If the new observing solution is better than the current solution, the new solution is accepted. Else, a non-improving move is accepted according to geometric cooling scheme (Ingber, 1995) used with success previously in the global optimization context (Özdamar and Demirhan, 2000):

$$AB(\Delta, t^j) = \exp(-(\Delta/t^j) \quad (16)$$

Where, AB is the probability of acceptance, Δ is the difference between $f(x_{j+1})$ and $f(x_j)$, and t_j is the temperature in iteration j.
If a randomly generated number between zero and one be less than AB, the deteriorating move is implemented. T_j is depending on the number of times that deteriorated solution has been accepted. It is decreased as follows:

$$t^{j+1} \leftarrow t^j \times 0.999 \quad (17)$$

3.2.6. Initial temperature

A suitable initial temperature is the one that results in an average increase of acceptance probability near to one. The value of initial temperature will clearly depend on the scaling of fitness and, hence, it should be problem-specific. Therefore, we first generate a large set of random solutions, then a standard division of them are calculated and is used to determine the initial temperature in the way that the acceptances probability of primary generations reach to 0.999.Consequently, the initial t_j is set to 1000based on some preliminary examinations.

3.2.7. Iteration limit at each temperature level

We define the set of m iterations as a ‘‘round’’. If the rate of change between mean fitness of two successive ‘‘round’’ of iterations remains constant within a pre-defined confidence interval, we conclude that the system has reached thermal equilibrium and reduce the temperature. Otherwise, we keep perturbing the solutions by creating neighbouring solutions until reaching equilibrium.

3.2.8. Solution representation

Each solution of our proposed algorithm is a super-matrix including two matrixes which the first one shows the sequence of cities and the second one illustrate the transporter types. Fig 1shows an example of our solution representation in which the first matrix is a $1 \times N$matrix and second one is a $N \times N$ matrix presented the selected transporting vehicle types. For instance, the salesman travels from city 6 to 2by transporter vehicle type 3, from city 2 to 4 by transporter vehicle type 5, from city 4 to 10 by transporter vehicle type 9, from city 10 to 5 by transporter vehicle type 5, from city 5 to 7 by transporter vehicle type 1, from city 3 to 8 by transporter vehicle type8, from city 8 to 1 by transporter vehicle type 6, and finally from city 1 to 9 by transporter vehicle type 7.

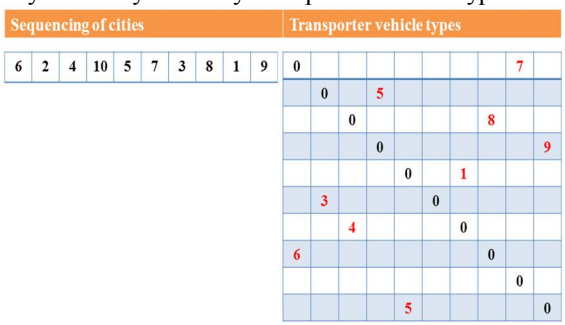

Fig. 1.An example of solution representation

3.2.9. Movement mechanism

We obtain a neighbour solutions by pairwise exchange heuristic called 2-opt exchange. It works by changing the position of two cities in tour at a same time.
The steps of our PSA algorithms (including PSA1, PSA2) are given in Figure2. These steps are similar to the steps of the algorithm given by Sahin et al. (2010) except that we are assuming different parameters and initial solutions for our PSAs algorithms; also we are incorporating different transporting vehicles and budget constraint.

The steps of PSA algorithms

Step 1: The distance, time, cost matrices and available budget for traveling are given as input data. The definitions of parameters for the parallel simulated annealing heuristic are as follows:
T_{in}= initial temperature, a = cooling rate, m= number of iteration which exhibit a round, $NIET$ = the number of trials to be performed at the same temperature value,el_{max}= maximum iteration number.
Step 2: Determine initial solution using mathematical techniques.
Step 3: Actualize the solution gained in step 2.
Step 4: Set temperature counter el = 1 (el: outer circle counter). Read initial solution (S_{in}) from file and calculate the total cost for the initial solution (C_{in})
Step 5: Set S_{best} (best solution) = S_c (current solution) = S_{in}, and also set C_{best}(best cost) = C_c(current cost) = C_{in} ,C_r(mean costs of the solutions in current round), C_{r-1} (mean costs of the solution in previous round). Make the iteration counter 1 at each temperature level: il = 1.
Step 6: Generate a neighbour solution (S_n) from the current solution as defined above. Then calculate the total cost of the neighbour solution (C_n).
Step 7: Check the cost of neighbour solution and. If the budget constraints are not satisfied, actualize it as defined above, then go to the next step.
Step 8: Calculate the change in objective function value:$D = C_n - C_c$.
Step 9: If (D< 0) go to Step 11; otherwise go to the next step.
Step 10: Generate a uniform random number (x) between [0, 1]. If x< ($e^{-D/T}$) go to the next step; otherwise go to step 13.
Step 11: Set to $S_c = S_n$ and $C_c = C_n$.
Step 12: If C_c<C_{best}, set $S_{best} = S_c$ and $C_{best} = C_c$.
Step 13:If (il<m), set $il = il + 1$, and go Step 6 otherwise go to the next step.
Step 14:Calculate“C_r”,if C_r-C_{r-1}dropsinto the confidence interval goto the next step otherwise set $C_{r-1} = C_r$ and go to Step 6.
Step 15: Set $el = el + 1$, $T_{el} + 1 = aTel$ and il = 1. (T_{el}: temperature atthe $elst$ iterations)
Step 16: If (el 6 el_{max}) go to Step 4; otherwise go to the next step.
Step 17: Stop algorithm and report the results.

Fig. 2.The steps of our PSA algorithms

To be able to compare the proposed algorithms fairly, all test problems with both algorithms were solved by the stopping criterion T_{min}=0.001, as defined previously.

4. Computational Results

All computational experiments were executed on an ASUS laptop with Pentium V PC, 2.4 GHz processors, Core i5, and 4 GByte of RAM. Also MATLAB software (Version 7.10.0.499, R2010a) was used to code the PSA, and the Minitab 16 software was used to tune the parameters. The LP's was solved by usingLINGO11.0 software. The proposed PSA algorithms (including PSA1, PSA2) are applied to ten test problems with 50, 100, 500 and 2000nodes. We assume 10 different transporter types and three types of total budget in hand. The allocation of total budget is carried out in this way: first, the average cost of traveling by each vehicle is calculated. Then, budget type 1 is found so that the salesman can travel 35% of path by the lowest-cost vehicles, 50% of path by the normal-cost vehicles, and 15% of path by the highest-cost vehicles. Budget type 2is found which salesman could travel 30% of path by the lowest-cost vehicles, 45% of path by the normal-cost vehicles, and 25% of path by the highest-cost vehicles. These percentages are 25%, 40%, and 35% for budget type 3. The distances between each city are integer number uniformly distributed from [0,1000]. The travel times are depended on the distances between each city and transporter types, and are calculated so that the higher-cost vehicles have the lower travel times, and reversely. The transporter costs are dependent on the transporter types and are computed so that the lower time consuming vehicles are more expensive, and reversely.

4.1. Parameter stuning

The initial parameters of our proposed PSA algorithms (including PSA1, PSA2) are the population size (N_{pop}) and the temperature decreasing rate (α). We used the Taguchi method of design of experiments (DOE) (Montgomery, 2005).In the Taguchi method, the results are transferred into a measure called signal to noise (S/N) ratio. The formulation of this ratio is difference for each objective(maximization or minimization). Eq (21) represent the (S/N) ratio for minimization objectives.

$$S/N = -10\log(1/n\sum_{i=1}^{n} y_i^2) \quad (18)$$

Which, n and y_i indicate number of replication and process response value at i'th replication.
The initial parameters including N_{pop} and α examined in three different levels and repeats for nine times, and we choose the orthogonal array L_9. After testing the different initial parameter levels and analysis of the gained results, the better initial parameter levels are determined. These values are depicted in Table 1. Also, the averaged S/N ratio for each factor level is shown in Fig 3.
According to the Fig 3, the best level of the population size (N_{pop}) is 200, and the best level of the temperature decreasing rate (α) is 0.95.

Table 1
The initial parameter levels

Parameters	Factor levels		
	1	2	3
N_{pop}	100	200	300
α	0.9	0.95	0.97

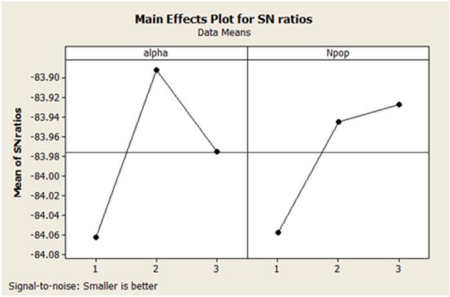

Fig. 3. Factor level trend of PSA algorithm

4.2. Analysis of results and comparisons

The computational results which are based on objective function and CPU time for TSPMT with 50, 100, 500 and 2000nodesand three assumed budget types for 10 different test problems are shown in Tables 2-5.
We used "Gap" as a percentage of deviation from the best quality solution of algorithm as below:

$$Gap = \frac{C_{Z^*} - C_{Z^{**}}}{C_{Z^*}} * 100 \quad (19)$$

Where $C_{Z^*}$is the solution obtained by PSA1 for a given instance, and $C_{Z^{**}}$is that obtained by PSA2 for the same instance.
In the columns 7 and 8 of Tables 2-5, Gap 1 means the percentage of deviation from the best quality solution of PSA1 and the value of the best quality solution obtained with PSA2. Gap 2 shows the percentage of deviation from the best quality CPU time of PSA1 and the value of the best quality CPU time obtained with PSA2. The optimal solutions of LINGO for small-sized problems with n=50 are depictedin column 9 of Table 1. In addition, we utilize the relative percentage deviation (RPD) to analyse the performance of our PSA2 algorithm as following formula:

$$RPD_i = \frac{Alg_m(i) - min_m(i)}{min_m(i)} * 100 \qquad i = 1, \dots, n_s \quad (20)$$

Where, $Alg_m$is objective function's value for a given algorithm, $min_m$is the best value of the objective function between both algorithms and n_s is number of small-sized or large-sized problems. Also, the results of each problem sizes by proposed PSA algorithms (including PSA1, PSA2)for all three budget types are shown in Figures 4-7.

Table 2
Detailed results of heuristic algorithms for problem size$n$=50

Test problems	Budget type	PSA1		PSA2		Gap 1 (%)	Gap 2 (%)	LINGO
		Solution	CPU time(s)	Solution	CPU time(s)			
1	1	4738.294	26.28	4675.304	25.57	1.33	2.70	4636.212
	2	4452.871	26.14	4301.258	26.04	3.40	0.38	4247.832
	3	4054.841	26.16	3976.152	26.75	1.94	-2.21	3921.901
2	1	4752.743	26.19	4481.725	25.59	5.70	2.29	4437.732
	2	4482.347	26.08	4286.142	26.13	4.38	-0.19	4240.441
	3	4157.849	26.00	4099.541	25.52	1.40	1.88	3927.934
3	1	5015.574	26.21	4645.639	26.54	7.38	-1.24	4602.579
	2	4528.284	26.02	4168.806	25.79	7.94	0.88	4004.501
	3	4219.879	26.07	4065.321	25.68	3.66	1.50	4019.021
4	1	4862.574	26.11	4784.174	26.05	1.61	0.23	4722.530
	2	4305.874	26.28	4237.382	26.42	1.59	-0.53	4176.267
	3	4157.594	26.44	4036.817	25.33	2.90	4.20	3978.834
5	1	4824.748	26.03	4544.507	25.73	5.81	1.15	4505.763
	2	4352.574	25.97	4288.054	25.14	1.48	3.20	4225.604
	3	4235.741	26.09	4121.093	26.42	2.71	-1.25	4066.452
6	1	4700.547	26.07	4679.620	25.93	0.45	0.54	4592.201
	2	4528.872	26.06	4479.255	25.85	1.10	0.81	4423.505
	3	4158.247	26.09	4012.004	26.21	3.52	-0.46	3956.625
7	1	4998.174	25.96	4669.445	25.45	6.58	1.96	4583.022
	2	4952.784	26.13	4565.092	25.65	7.83	1.84	4507.490
	3	4784.541	26.51	4333.351	25.07	9.43	5.43	4288.534
8	1	4862.429	26.18	4655.727	26.16	4.25	0.08	4607.804
	2	4457.189	26.02	4228.762	25.55	5.12	1.81	4169.522
	3	4325.281	26.13	4126.737	26.11	4.59	0.08	4064.540
9	1	4862.478	26.16	4442.481	26.26	8.64	-0.38	4405.672
	2	4497.682	26.36	4393.405	25.45	2.32	3.45	4321.307
	3	4008.254	26.05	3864.898	25.79	3.58	0.10	3824.480
10	1	4896.438	26.03	4614.662	25.90	5.75	0.50	4567.957
	2	4532.128	26.22	4295.521	26.14	5.22	0.30	4254.146
	3	4212.849	26.17	4091.833	25.79	2.87	1.45	4037.194
Average	1	4851.400	26.122	4619.328	25.918	4.78	0.78	4566.147
	2	4509.061	26.128	4324.368	25.816	4.10	1.19	4257.062
	3	4231.508	26.171	4072.775	25.867	3.75	1.16	4008.552

Table 3
Detailed results of heuristic algorithms for problem size n=100

Test problems	Budget types	PSA1		PSA2		Gap 1 (%)	Gap 2 (%)
		Solution	Time(s)	Solution	Time(s)		
1	1	8854.254	90.33	8542.704	88.25	3.52	2.30
	2	8120.542	89.26	8032.409	87.06	1.09	2.46
	3	7994.769	91.11	7285.002	89.34	8.88	1.94
2	1	9234.984	89.60	8300.946	90.32	10.11	-0.80
	2	7999.175	89.22	7553.128	88.45	5.58	0.86
	3	7587.863	91.21	6978.414	90.51	8.03	0.77
3	1	9672.263	89.28	9115.156	88.05	5.76	1.38
	2	8969.821	90.05	8147.821	87.78	9.16	2.52
	3	7854.222	91.87	7743.594	89.69	1.41	2.37
4	1	8984.730	89.42	8319.401	89.05	7.40	0.41
	2	8463.818	90.99	8198.627	88.40	3.13	2.85
	3	7951.004	91.43	7950.165	87.32	0.01	4.50
5	1	9348.261	89.52	8503.745	87.92	9.03	1.79
	2	8360.679	91.73	8038.297	90.34	3.86	1.52
	3	7508.705	90.84	7213.896	88.37	3.93	2.72
6	1	8552.777	89.44	7700.083	89.04	9.97	0.45
	2	7441.408	91.58	7280.998	88.63	2.16	3.22
	3	7267.651	95.85	7192.328	89.51	1.04	6.61
7	1	8472.267	89.70	7809.362	89.73	7.82	-0.26
	2	7818.410	92.09	7564.429	90.34	3.25	1.90
	3	7394.579	96.24	7047.321	89.67	4.70	6.83
8	1	8564.057	89.55	8196.377	89.79	4.29	-0.27
	2	7815.675	91.22	6990.574	89.46	10.56	1.93
	3	7481.610	97.86	6839.151	90.31	8.59	7.71
9	1	9021.420	89.76	8125.251	89.56	9.93	0.22
	2	8456.124	91.53	7585.321	89.19	10.30	2.56
	3	7913.856	95.23	7243.681	90.37	8.47	5.10
10	1	8240.684	89.70	8004.254	90.34	2.87	-0.71
	2	8005.743	91.44	7784.265	91.22	2.77	0.24
	3	7607.217	94.99	6952.251	91.80	8.61	3.36
Average	1	8894.570	89.630	8261.728	89.205	7.11	0.47
	2	8145.140	90.911	7717.587	89.087	5.25	2.01
	3	7656.148	93.663	7244.580	89.689	5.38	4.24

Table 4
Detailed results of heuristic algorithms for problem size n=500

Test problems	Budget types	PSA1		PSA2		Gap 1 (%)	Gap 2 (%)
		Solution	Time(s)	Solution	Time(s)		
1	1	58996.639	418.05	56956.555	401.56	3.46	3.94
	2	57330.802	413.76	55383.779	408.38	3.40	1.30
	3	55821.389	414.52	52382.183	403.53	6.16	2.65
2	1	58013.351	415.34	56228.680	398.30	3.08	4.10
	2	56957.230	413.40	55184.962	405.77	3.11	1.85
	3	54538.999	417.16	51693.657	401.76	5.22	3.69
3	1	59754.061	413.52	57438.586	402.94	3.88	2.56
	2	56150.643	419.50	55346.940	410.25	1.43	2.21
	3	52125.111	414.71	51190.466	408.83	1.79	1.42
4	1	62574.293	418.70	58529.637	409.05	6.46	2.30
	2	61574.352	411.69	56964.267	401.52	7.49	2.47
	3	57514.529	416.73	54844.911	407.69	4.64	2.17
5	1	61824.617	419.60	59785.636	401.16	3.30	4.39
	2	60579.253	413.70	58257.889	399.24	3.83	3.50
	3	58642.867	415.67	55552.123	404.31	5.27	2.73
6	1	65984.279	420.60	61278.957	408.84	7.13	2.80
	2	63583.418	414.70	60513.453	407.36	4.83	1.67
	3	60745.513	411.88	57915.796	399.83	4.66	2.93
7	1	61826.739	418.31	58537.871	410.45	5.32	1.88
	2	59543.848	423.09	55340.669	407.78	7.06	3.70
	3	56857.219	420.97	54506.916	401.73	4.13	4.57
8	1	66531.349	421.22	59432.375	407.37	10.67	3.29
	2	57720.603	415.13	56746.706	403.31	1.69	2.85
	3	54321.201	419.23	51257.414	405.40	5.64	3.30
9	1	62184.365	422.34	58009.065	407.74	6.71	3.46
	2	56279.415	418.11	53283.216	406.07	5.32	2.88
	3	53841.569	415.42	50957.414	409.39	5.36	1.45
10	1	68318.946	420.95	64669.425	408.83	5.34	2.88
	2	66843.509	419.15	63843.890	403.16	4.49	3.81
	3	64851.397	423.27	62557.414	409.77	3.54	3.19
Average	1	62600.86	418.863	59086.68	405.624	5.61	3.16
	2	59656.31	416.223	57086.58	405.284	4.31	2.63
	3	56925.98	416.956	54285.83	405.224	4.64	2.81

Table 5
Detailed results of heuristic algorithms for problem size$n$=2000

Test problems	Budget types	PSA1		PSA2		Gap 1 (%)	Gap 2 (%)
		Solution	Time(s)	Solution	Time(s)		
1	1	409254.254	1012.34	386457.287	981.47	5.57	3.05
	2	391002.916	1013.63	361577.403	980.67	7.53	3.25
	3	368525.254	1008.49	349463.622	984.36	5.17	2.39
2	1	375424.254	1018.85	363842.924	984.59	3.08	3.36
	2	354302.265	1015.71	332851.600	980.74	6.05	3.44
	3	345698.254	1022.04	327596.928	987.07	5.24	3.42
3	1	412324.254	1020.07	385143.538	985.85	6.59	3.35
	2	389989.166	1016.87	361546.647	983.38	7.29	3.29
	3	363254.254	1010.05	339546.647	991.27	6.53	1.86
4	1	375258.574	1009.11	359527.816	989.33	4.19	1.96
	2	359472.477	1014.74	340321.422	983.64	5.33	3.06
	3	339245.254	1020.58	324627.999	985.16	4.31	3.47
5	1	391251.221	1010.47	378245.284	988.38	3.32	2.19
	2	377094.136	1016.67	352285.432	991.62	6.58	2.46
	3	347258.265	1022.85	328862.401	984.43	5.30	3.76
6	1	368784.448	1009.82	354596.892	988.39	3.85	2.12
	2	359640.674	1025.53	349527.826	985.70	4.18	3.88
	3	348825.368	1013.21	330498.095	991.96	5.25	2.10
7	1	364173.846	1008.58	350507.977	989.47	3.75	1.89
	2	353224.093	1011.65	335887.321	985.31	4.91	2.60
	3	349872.849	1018.63	322141.029	992.44	5.07	2.57
8	1	381283.942	1009.31	368573.514	981.07	3.33	2.80
	2	364130.176	1017.52	341028.742	984.65	6.34	3.23
	3	349571.927	1022.83	329434.337	990.06	5.76	3.20
9	1	399541.973	1012.81	382498.551	983.31	4.27	2.91
	2	375159.266	1018.27	356358.254	985.75	5.01	3.19
	3	354581.854	1020.34	332883.913	990.38	6.12	2.94
10	1	370058.489	1009.80	349854.286	981.76	5.46	2.78
	2	358541.592	1021.98	334570.325	992.27	5.21	2.91
	3	341573.534	1014.33	329899.496	985.29	3.42	2.86
Average	1	384735.5	1012.116	367924.8	985.362	4.37	2.64
	2	368255.7	1017.257	346595.5	985.373	5.88	3.13
	3	350840.7	1017.335	331495.4	988.242	5.51	2.86

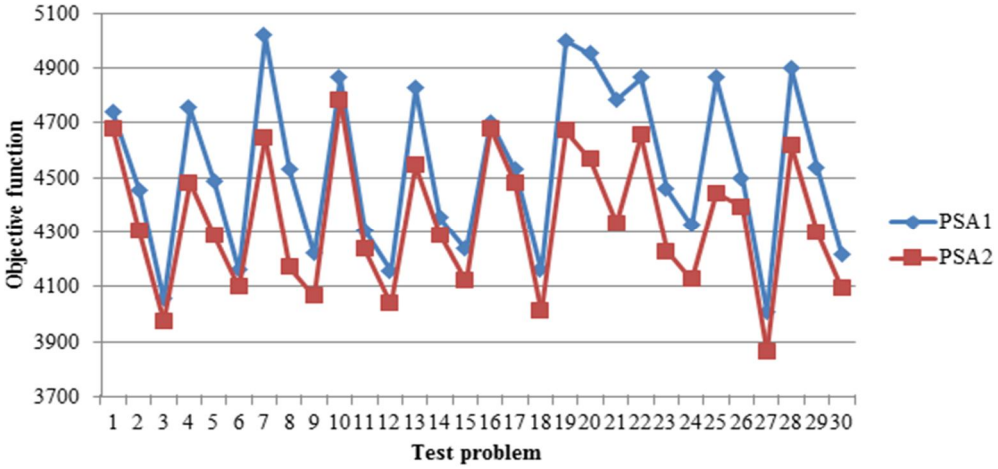

Fig. 4.Objective function of 30 test problems for n=50

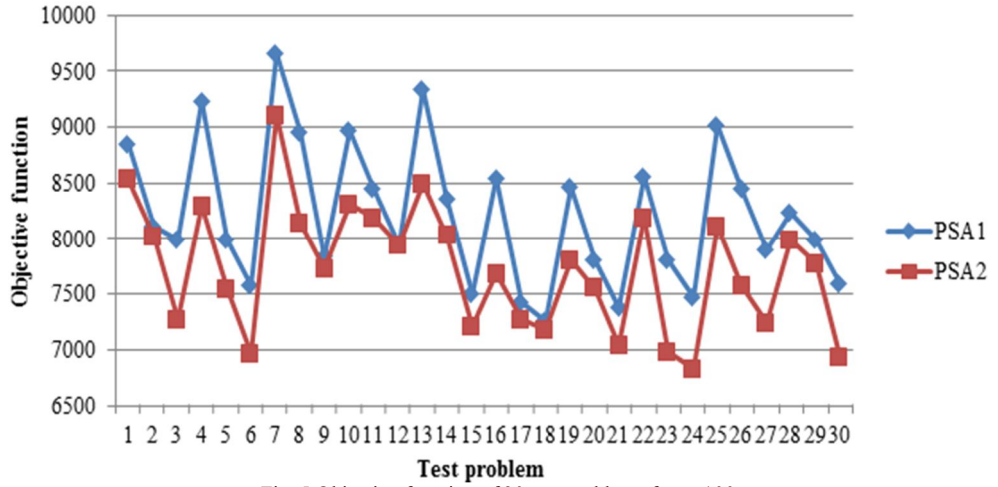

Fig. 5.Objective function of 30 test problems for n=100

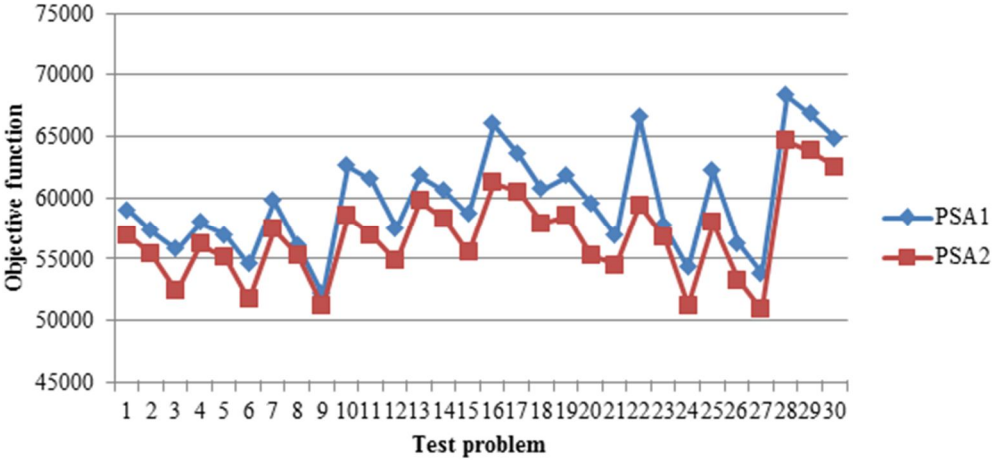

Fig. 6.Objective function of 30 test problems for n=500

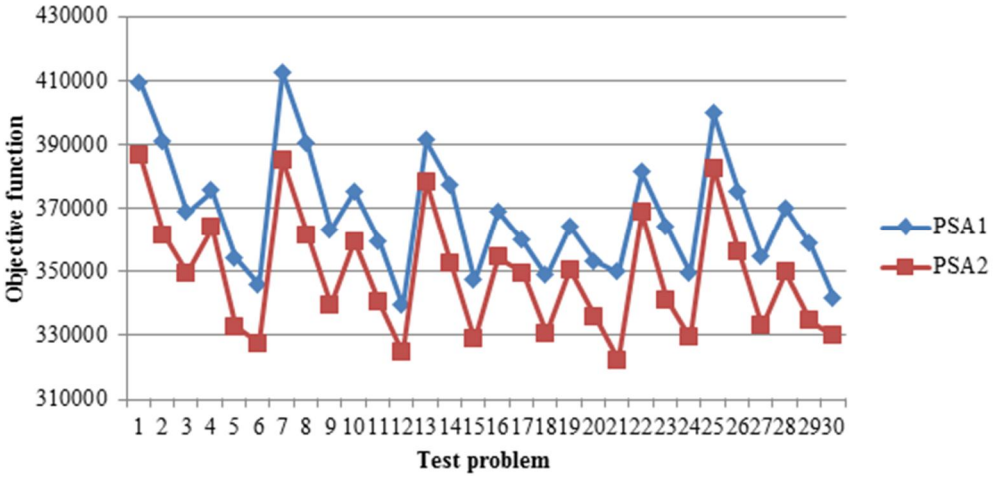

Fig. 7.Objective function of 30 test problems for n=2000

Table 2 shows the results for small-sized problems. According to the computational results of PSA algorithms(including PSA1, PSA2), in Table 2 with n=50, PSA2has achieved better quality in solutions with 4.210% average deviation of the gap factor in comparison to PSA1. Table 3 gives the results for medium-sized problems. In Table 3 with n=100, PSA2 has obtained better results of solutions with 5.913% average deviation of gap factor than PSA1. Table 4 and 5 show the results for large-sized problems. In Table 4 with n=500, PSA2 reaches to better quality solutions with 4.853% of average gap factor deviation than PSA1, and finally in Table 5 with n=2000, PSA2 has better quality solutions with 5.253% average deviation of gap factor rather than PSA1.Another important factor regarding the proposed algorithm is the "CPU time" comparisons. As the results of average CPU time, in Table 2 with n=50 show, PSA2 has found better times with 1.043% average deviation of gap factor against PSA1. In Table 3 with n=100, PSA2 has reached to better times with 2.240% average deviation of gap factor than PSA1. In Table 4 with n=500, PSA2 has achieved better times with 2.867% average deviation of gap factor than PSA1, and, finally, in Table 5 with n=2000, PSA2 has obtained better times with 2.876% average deviation of gap factor against PSA1.

Also, according to the average RPD (ARPD) comparisons of two proposed algorithms, PSA2 has better quality with 4.964, 7.174, 5.707, 5.982 deviations against PSA1 for n=50, 100, 500, and 2000, respectively. Also in terms of CPU time, PSA2 has better solutions quality considering ARPD with 1.09, 2.383, 3.005, 2.982 deviations against PSA1 for n=50, 100, 500, and 2000, respectively. Figures 8 and 9 show the ARPD of objective function and CPU time of proposed PSA algorithms, respectively. It should be noted that the newly-developed TSPMT model is more complicated than the traditional versions due to the ability of selecting different transporting vehicles. Just on each route, we have a large increase in the number of variables because of variety in transporting vehicle types. Therefore, the CPU time of around 1000 seconds for such a large-sized problem is rational.

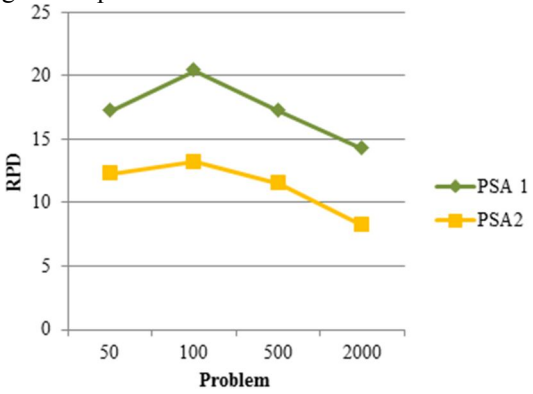

Fig. 8.ARPD of objective function of PSA heuristics

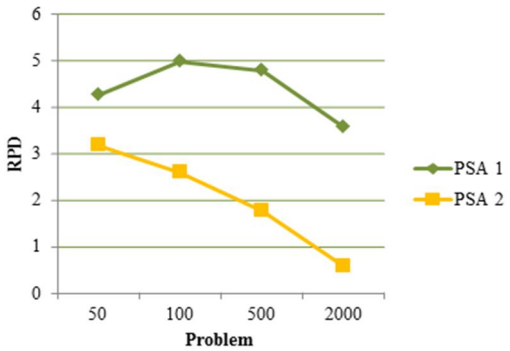

Fig. 9.ARPD of CPU time of PSA heuristics

In addition, we use the 95% confidence intervals T-test to compare the near optimal best solutions of objective function and CPU time. The results of these comparisons are depicted in Figures 10 and 11. We can infer from these statistical tests, because of the p-value (0.000) is smaller than α (0.05), there is a significant difference between PSA1 and PSA2 in terms of objective function and CPU time.

Two-sample T –test for objective functions of PSA1 and PSA2

	N	Mean	StDev	SE Mean
C1	120	17.26	8.15	0.74
C3	120	11.30	7.14	0.65

Difference = mu (PSA1) - mu (PSA2)
Estimate for difference: 5.957
95% CI for difference: (4.008, 7.906)
T-Test of difference = 0 (vs not =): T-Value = 6.02 P-Value = 0.000 DF = 233

Fig. 10.Statistical results of T-test of objective function

Two-sample T –test for CPU times of PSA1 and PSA2

	N	Mean	StDev	SE Mean
C2	120	4.40	1.53	0.14
C4	120	2.03	1.50	0.14

Difference = mu (PSA1) - mu (PSA2)
Estimate for difference: 2.365
95% CI for difference: (1.979, 2.751)
T-Test of difference = 0 (vs not =): T-Value = 12.07 P-Value = 0.000 DF = 233

Fig. 11.Statistical results of T-test of CPU time

Figures 12 and 13 show the 95% confidence intervals of RPD for objective functionand CPU time indexes, respectively.For each problem sizes, we can see that PSA2 gives better results than PSA1 in terms of both objective function and CPU time, especially in large-sized problems.

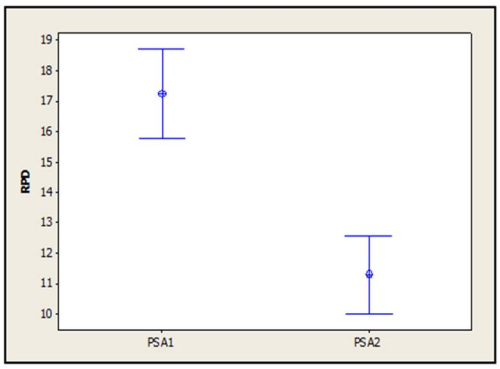

Fig. 12.The 95% confidence intervals of RPD of objective function

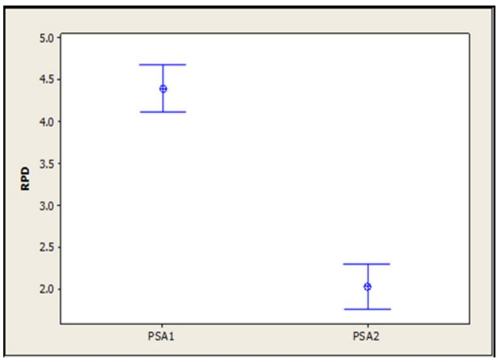

Fig. 13.The 95% confidence intervals of RPD of CPU time

5. Conclusions and suggestions for future research

In this study, a new version of the well-known traveling salesman problem was developed in which the salesman has some options to select the best vehicle type in each city considering a predefined budget. This new version of TSP which is the main contribution of the research named TSPMT which is more compatible to the real world problems than the traditional versions. First, the mathematical programming model of TSPMT was presented, and then the PSA2 with a new coding scheme was built to solve the different size of problems and compared it with PSA1. So, 50 numerical examples with three budget types were developed to test the performance of the proposed algorithm. According to the obtained results, PSA2 yields better solutions in comparison to PSA1 in terms of objective function and time consumption, especially in large-sized problems. For future researches, developing a new solution methodology such as a new hybrid algorithm or a new population-based algorithm in which the initial solutions are obtained in a heuristic manner can be investigated.The budget constraint that we used in our test problems restricts the budget for a whole traveling path. One attempt is to solve problems with different budget constraints. For example, separate budget constraints can be applied for each city. Another aspect deserving future efforts is to consider that the salesman should travel only by the existing transporting vehicles that are available in each city with limited budget.

6. References

[1] Albayrak, M., Allahverdi, N. (2011).Development a new mutation operator to solve the Traveling Salesman Problem by aid of Genetic Algorithms, Expert Systems with Applications, 38, 1313-1320.

[2] Balas,E., Christofids,N. (1981).A restricted lagrangian approach to the traveling salesman problem, Mathematical Programming, 21, 19–46.

[3] Bianchi, L. (2006). Ant colony optimization and local search for the probabilistic traveling salesman problem: a case study in stochastic combinatorial optimization, Ph.D. Thesis, University of Libre de Bruxelles, Brussels, Belgium.

[4] Bonomi, E., Lutton, J. L. (1984).The *N*-city traveling salesman problem: statistical mechanics and the metropolis algorithm, SIAM Review, 26, 551-568.

[5] Bontoux, B., Feillet, D. (2008). Ant colony optimization for the traveling purchaser problem, Computers and Operational Research, 35: 628-637.

[6] Bontoux, B., Artigues, C., Feillet, D. (2010).A memetic algorithm with a large neighbourhood crossover operator for the generalized traveling salesman problem, Computers and Operations Research, 37, 1844-1852.

[7] Branke, J., Guntsch, M. (2004). Solving the probabilistic TSP with ant colony optimization, Journal of Mathematical Model, Algorithms 3.

[8] Braun, H. C. (1991).On solving traveling salesman problem by genetic algorithm, in: H. P Schwefel, R. Manner (Eds.), Parallel problem solving from nature, in: Lecture notes in Computer Science, 496, 129-133.

[9] Crowder, H., Padberg, M. W. (1980). Solving large-scale symmetric traveling salesman problems to optimality. Management Science, 26, 495-509.

[10] Dantzig, G. B., Fulkerson, D. R., Johnson,S. M. (1954). Solution of a large-scale traveling salesman. Operations Research, 2, 393–410.

[11] Ergan,Ö.,Orlin,L. B. (2006).A dynamic programming methodology in very large scale neighbourhood search applied to the traveling salesman problem, Discrete Optimization, 3, 78-85.

[12] Goldberg, D. E., Lingle, R. (1985).Thetraveling salesman problem. Proceedings of the International Conference on Genetic Algorithms and their Applications, Carnegie-Mellon University, 154-159.

[13] Goldberg, D. E. (1989). Genetic algorithms in search, optimization and machine learning, Addison-Wesley, New York.

[14] Grötschel, M., Holland,O.(1991).Solution of large-scale symmetric traveling salesman problems. Mathematical Programming, 51, 141-202.

[15] Glover, F. (1990). Artificial intelligence, heuristic frameworks and tabu search. Managerial and Decision Economics, 11, 365-375.

[16] Grefenstette, J. J., Gopal,R., Rosimaita, B., Van Gucht, D.(1985). Genetic algorithms for the traveling salesman problem, in: Proceedings of the International Conference on Genetic Algorithms and their Applications, 160-168.

[17] Held, M., Karp, R. M. (1971).The traveling salesman problem and minimum spanning trees. Mathematical Programming, 1, 6–25.

[18] Hopfeild, J. J., Tank, D. W. (1985).Neural computation of decision in optimization problems.Biological cybernetic, 52-60.

[19] Ingber, L.(1995). Adaptive simulated annealing (ASA): Lessons Learned. To appear in control and cybernetics, (Polish Journal).

[20] Jog, P., Suh, J. Y., Van Gucht, D. (1989).The effect of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem, in: Proceeding of the Third International Conference on Genetic Algorithms and their Applications, 110-115.

[21] Jun-man, K., Yi, Z. (2012). Application of an improved ant colony optimization on generalized traveling salesman problem, Energy Procedia, 17, 319-325.

[22] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P.(1983). Optimization by simulated annealing. Science, 220, 671-680.

[23] Kirkpatrick, S., Toulouse, G. (1985). Configuration space analysis of traveling salesman problem. Journal of Physique, 46, 1277-1292.

[24] Knox, J. (1989).The application of tabu search to the symmetric traveling problem. Ph.D Dissertation, University of Colorado.

[25] Kuroda, M., Yamamori,K., Munetomo, M., Yasunaga, M., Yoshihara, I. (2010).Development of a novel crossover of hybrid genetic algorithms for large-scale traveling salesman problems. Artificial Life Robotics, Springer, 15, 547-550.

[26] Lam, J. (1988).An efficient simulated annealing schedule. Ph.D Dissertation, Department of computer and science, Yale University.

[27] Lawler, E. L., Lenstra,J. K., RinnooyKan, A. H. G., Shmoys, D. B.(1985). The traveling salesman problem. A guided tour of combinatorial optimization Wiely and Sons, New York.

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

[29] Liu, Y-H. (2008). Diversified local search strategy under scatter search framework for the probabilistic traveling salesman problem, European Journal of Operational Research, 191, 332-346.

[30] Marinakis, Y., Marinaki, M. (2010) A hybrid multi-swarm particle optimization algorithm for the probabilistic traveling salesman problem, Computers and Operations Research, 37, 432-442.

[31] Martin, O., Otto, S. W., Felten, E. W.(1991) Large scale markov chains for the traveling salesman problem, Complex system 2, 299-326.

[32] Majumdar, J., Bhunia, A. K. (2011).Genetic algorithm for asymmetric traveling salesman problem withimprecise travel times, Journal of Computational and Applied Mathematics, 235, 3063-3078.

[33] Mavrovouniotis, M., Yang, S. (2013).Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors, Applied Soft Computing, 13, 4023-4037.

[34] Miller, C. E., Tucker, A. W., Zemlin, R. A. (1960).Integer programming formulation of traveling salesman problems.Journal of ACM, 7, 326–9.

[35] Montgomery, D. C. (2005). Design and analysis of experiments. Arizona, John Wiley and Sons.

[36] Nagata, Y., Soler, D. (2012).A new genetic algorithm for the asymmetric traveling salesman problem, Expert Systems with Applications, 39, 8947-8953.

[37] Onbasoglu, E., Özdamar, L. (2001). Parallel simulated annealing in global optimization. Journal of Global Optimization, 19, 27-50.

[38] Özdamar, L., Demirhan, M. (2000). Experiments with new stochastic global optimization search techniques. Computer and OR27: 841-865.

[39] Padberg, M., Rinaldi, G. (1980).Optimization of a 532 city symmetric traveling salesman problem by branch and cut. Operations Research, 6, 1-7.

[40] Pedro, O., Saldanha, R., Camargo, R. (2013).A tabu search approach for the prize collecting traveling salesman problem, Electronic Notes in Discrete Mathematics, 41, 261-268.

[41] Rego, C., Gamboa, D., Glover, F., Osterman, C. (2011). Traveling salesman problem heuristics: Leading methods, implementations and latest advance. European Journal of Operation Research, 211, 427–441.

[42] Sahin, R., Ertoĝral, K., Türkbey, O. (2010). A simulated annealing heuristic for the dynamic layout problem with budget constraint. Computers and Industrial engineering, 59, 308-313.

[43] Whitley, D., Starkweather,T.(1989). Scheduling problems and traveling salesman: the genetic edge recombination operator, in: Proceedings of the Third International Conference on Genetic Algorithms and their Applications, 133-140.

[44] Xing L. N., Chen, Y. W., Yang, K. W., How, F., Shen, X. S., Cai, H. P.(2008). A hybrid approach combining an

improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem. Engineering Applications of Artificial Intelligence, 21, 1370-1380.

[45] Zhang, J., Feng, X., Zhou, B., Ren, D. (2012).An overall-regional competitive self-organizing map neural network for the Euclidean traveling salesman problem, Neurocomputing, 89, 1-11.