International Journal of Computational Intelligence Systems, Vol. 7, No. 5 (October 2014), 827-852

A Self-Adaptive Heuristic Algorithm for Combinatorial Optimization Problems

Cigdem Alabas-Uslu*
Marmara University, Department of Industrial Engineering, 34722 Istanbul, Turkey
cigdem.uslu@marmara.edu.tr

Berna Dengiz
Baskent University, Department of Industrial Engineering, 06530 Ankara, Turkey
bdengiz@baskent.edu.tr

Received 22 March 2013

Accepted 18 January 2014

Abstract

This paper introduces a new self-tuning mechanism to the local search heuristic for solving of combinatorial optimization problems. Parameter tuning of heuristics makes them difficult to apply, as parameter tuning itself is an optimization problem. For this purpose, a modified local search algorithm free from parameter tuning, called Self-Adaptive Local Search (SALS), is proposed for obtaining qualified solutions to combinatorial problems within reasonable amount of computer times. SALS is applied to several combinatorial optimization problems, namely, classical vehicle routing, permutation flow-shop scheduling, quadratic assignment, and topological design of networks. It is observed that self-adaptive structure of SALS provides implementation simplicity and flexibility to the considered combinatorial optimization problems. Detailed computational studies confirm the performance of SALS on the suit of test problems for each considered problem type especially in terms of solution quality.

Keywords: Metaheuristics, Combinatorial optimization, Parameter tuning, Adaptive parameter.

[*] Corresponding author

1. Introduction

Due to the practical and the theoretical importance of combinatorial optimization problems, interest in research to develop exact and heuristic algorithms has been evolved consistently. The run time of exact algorithms often increases exponentially with the instance size and only small or moderate-sized problems can be solved. Therefore, the use of heuristics to solve larger instances has been unavoidable. Particularly, the literature has been increasingly enlarged by metaheuristic approaches since the late 1980s. The survey carried out by Blum and Roli[1] and the book by Glover and Kochenberger[2] give the most popular metaheuristics from a conceptual point of view and outlines the details of different components and concepts.

Metaheuristics are controlled by a set of parameters. This set has a significant impact on the solving progress since parameters drive exploitation and exploration rate of search space. Thus, through the search process a solution is obtained with high quality. Parameters are the re-configurable parts of a metaheuristic algorithm that can be manipulated to alter the performance of the heuristic. Therefore, the best combination of parameter values is a crucial task. This task is generally called parameter optimization, parameter tuning or parameter setting. A careful selection of the best parameter set requires either a deep knowledge of the problem structure or a lengthy trial-and-error process. Tuning a set of parameters to achieve robust and high performance of the metaheuristic is a tedious and time consuming process. Adenso-Diaz and Laguna[3] state that about 10% of the total time dedicated to designing and testing of a new heuristic is spent for development, and the remaining 90% is consumed by fine-tuning of parameters. Today the operational research literature includes the large number of sophisticated metaheuristics which are considerably effective and efficient for the most combinatorial problems. Nevertheless, the most of them still are influenced by tediousness of parameter optimization.

Silberhorz and Golden[55] state that metaheuristics with a low degree of complexity have a number of advantages such as being simple to implement in an industrial setting, being simple to re-implement by researchers, and being simpler to explain and analyze. Meanwhile, as the heuristics get complicated, the number of parameters increases in general. Therefore a meaningful metric to measure complexity of the heuristics becomes the number of parameters used in the algorithm.

The best parameter set is usually re-determined before the run considering application area, size or input data of each individual instance. Many researchers tune the parameters applying different reasonable values and then select the combination which generates the best performance of the algorithm. There have been a number of studies which propose systematic methods to find the best parameter set for considered algorithm. While Barr et al.[4] use experimental design technique, Adenso-Diaz and Laguna[3] combine factorial experimental design with a local search mechanism.

An alternative way to tuning parameters beforehand is by controlling them throughout the run. Heuristics which are managed by this way are generally called adaptive, reactive or self-tuning heuristics. This kind of heuristics utilize differing forms of feedback information to perform a learning process of the parameter combination during the search. Self-tuning heuristics are achieved for evolutionary algorithms earlier than local search based algorithms. Eiben et al.[5] present a comprehensive study to classify parameter control methods for evolutionary algorithms and survey various forms of control methods. The pioneering attempt to develop a self-tuning mechanism for the local search based metaheuristics is the reactive tabu search by Battiti and Tecchiolli[6]. Today, numerous studies describing different dynamic parameter structures can be cited. For instance, scatter search by Russell and Chiang[7], threshold accepting by Tarantilis et al.[8, 9], record-to-record travel by Li et al.[10], and reactive tabu search by Osman and Wassan[11] are among the recent metaheuristics with dynamic parameters proposed for the vehicle routing problems.

In this study, a self-adaptive local search method, named SALS, is proposed. SALS algorithm has only one parameter notated by Θ and called *acceptance parameter*. Θ is obtained and updated dynamically throughout the search process. Thus, the effectiveness of the algorithm is improved using the response surface information of the problem instance and the performance measure of the algorithm. The most important advantage of SALS is that the algorithm does not need additional time and specialization to manage parameter optimization. Therefore, SALS is suggested as a heuristic with a low degree of complexity. We aim to show that SALS is able to generate very good solutions to combinatorial optimization problems without any tuning effort by applying it to problems selected from different application areas, namely, the classical vehicle routing (VRP), permutation flow shop scheduling (PFSP), quadratic assignment (QAP), and topological design of computer networks (TDP), problems.

Remainder of this paper is organized as follows. The structure of SALS algorithm is explained in Section 2. Implementations of SALS and tabu search (TS), simulated annealing (SA), record-to-record-travel algorithms (RRT) on the selected problems are given in Section 3. Section 4 contains comparison of SALS with TS, SA, and RRT algorithms on the test problems. Section 4 also includes another comparative study to demonstrate the effectiveness of SALS with respect to the some heuristic algorithms proposed in the VRP, PFSP, QAP, and TDP literatures. Finally, the last section presents the conclusions of this study.

2. Description of the Self-Adaptive Local Search Algorithm

SALS is a local search algorithm. The algorithm starts with any initial solution $\mathbf{X}_z$ as a current solution and searches the solution space iteratively. Vector of $\mathbf{X} = [x_1, x_2, \ldots, x_n]$ represents decision variables of considered problem. At iteration i, a neighbor solution $\mathbf{X'}$ is selected randomly from the neighborhood of the current solution $\mathbf{X}$. $\mathbf{X'}$ is recorded as the new current solution if the following condition is satisfied for a minimization problem:

If $f(\mathbf{X'}) \leq \Theta f(\mathbf{X})$ then $\mathbf{X} \leftarrow \mathbf{X'}$

Here, $f(\mathbf{X})$ is the objective function value of the solution $\mathbf{X}$ at iteration i where Θ is the self-adaptive parameter of SALS. The search process around the current solution, $\mathbf{X}$, is repeated until obtaining of an acceptable neighbor solution, $\mathbf{X'}$. The algorithm is progressed to the next iteration whenever a new current solution is recorded ($\mathbf{X} \leftarrow \mathbf{X'}$). If the total number of rejected neighbors reaches the neighborhood size of the current solution, $|\mathrm{N}(\mathbf{X})|$, at any iteration i, Θ is adjusted according to "$\Theta \leftarrow \Theta + \alpha_1\alpha_2$" only for the current iteration. Learning process of parameter Θ is based on two criteria: Quality of the best solution and number of improved solutions obtained during the search process. Hence, α_1 and α_2, given by equations 1-2, are introduced to measure the quality and the count of the searched solutions, respectively. Where, $\mathbf{X}_b^{(i)}$ is the best solution observed until iteration i, $\mathbf{X}_z$ is the initial solution, $\mathrm{C}(\mathrm{L}^{(i)})$ is the number of improved solutions obtained until iteration i:

$$\alpha_1 = \frac{f(\mathbf{X}_b^{(i)})}{f(\mathbf{X}_z)} \tag{1}$$

$$\alpha_2 = \frac{\mathrm{C}(\mathrm{L}^{(i)})}{i} \tag{2}$$

$$\Theta = 1 + \alpha_1\alpha_2 \tag{3}$$

The number of improved solutions until iteration i, is counted by $\mathrm{C}(\mathrm{L}^{(i)}) \leftarrow \mathrm{C}(\mathrm{L}^{(i)}) + 1$, if $f(\mathbf{X'}) < f(\mathbf{X}_b^{(i)})$ for an accepted neighbor solution $\mathbf{X'}$. SALS assumes that $f(\mathbf{X}) > 0$, for the whole solution space. Decreasing values of α_1 represent that solution quality of the best solution is improved comparing to the initial solution. On the other hand, constantly decreasing values of α_2 indicate flat regions of the solution space, while fluctuating values of that may indicate the regions with many local minima. Θ utilizes both α_1 and α_2 calculated through the search process adaptively as given in equation 3. Parameter Θ determines the borders of the search region in terms of objective function value surrounding the current solution $\mathbf{X}$. During the iterations of SALS, the measures of α_1 and α_2 are updated by equations 1 and 2, respectively, and thereby Θ is recomputed at each iteration using equation 3. Θ tends to take smaller values (approaching to 1) during the last part of the search. It is expected that while Θ approaches to 1, the search is forced to find better solutions. Figure 1 depicts the decrease of relative deviation from the reference solution accompanied by parameter Θ for selected instances from the VRP, PFSP, QAP, and TDP. Furthermore, changing of parameter Θ with respect to the number of iterations for these problems is shown in Figure 2 (in this figure initial iterations of the search process are ignored to provide clear visibility of the remainder iterations). As seen from the figures, the self-adaptive structure provides that the values of Θ alter depending on the application area. On the other hand, Θ exponentially decreases as the number of iterations increases for all problem types.

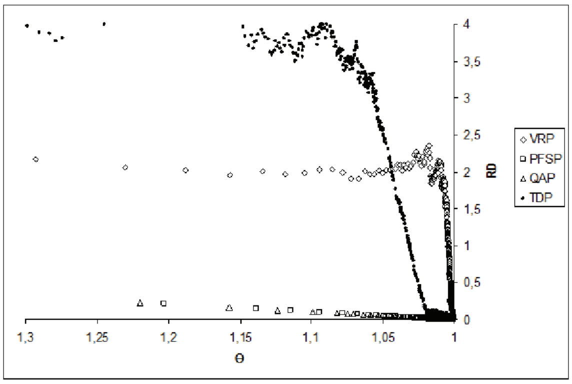


Fig. 1 Changing of Θ according to relative deviation from the best solution

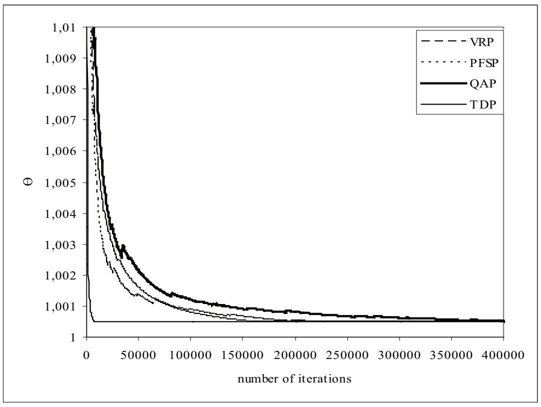


Fig. 2 Changing of Θ according to number of iterations

An experimental study is carried out to show the effectiveness of the self-adaptive of Θ by comparing three fixed levels, such as 1.0015, 1.0025, and 1.0035, so that they produce reasonably good results in the preliminary experiments. The VRP, PFSP, QAP, and TDP benchmark problem sets taken from Christofides and Elion[12], Taillard[13], Skorin-Kapov[14] and Altiparmak[15], respectively, are used for the experimental analysis. Problem instances are selected randomly for each size to be able to get a representative subset of the associated benchmarking set and classified as small, moderate, and large size problems. SALS is run with considered levels of Θ for each instance. The algorithm is allowed to run until a pre-determined number of solutions met. Table 1 shows the average deviations from the best known solutions (abbreviated as *ARD*) and also the standard deviation of the deviations obtained over the 10 runs. Totally 30 runs are made for each problem type at each Θ level. When Θ is equal 1.0015 the SALS algorithm generally yields better results (marked by *italic* fonts) than other fixed levels. However, it is easily seen that it is not robust against problem type and problem size. On the other hand, the SALS algorithm with self-adaptive determined Θ gives better results for all problem types and sizes. This means that while the SALS algorithm with fixed Θ value needs parameter tuning for each problem type, there is no need to spend more effort for the tuning of parameter Θ which is determined dynamically using self adaptive structure. As a result we can say that self-adaptive Θ generates the superior results (marked by **bolt** fonts) than those with all fixed levels except only three cases. Self-adaptive Θ also outperforms all of the fixed levels in terms of average results over the problem sizes. As seen from the Table 1, self-adaptive Θ generates the smallest standard deviation of *ARD* for QAP and TDP, while Θ = 1.0015 gives the smallest standard deviation for VRP and Θ = 1.0035 for PFSP. To statistically compare the levels of Θ, the Wilcoxon Signed Rank Test is applied to data gathered from the experimentation of SALS with different levels of Θ on each problem type under consideration. The Wilcoxon Signed Ranks test is designed to test a hypothesis about the mean of a population distribution. This test does not require the assumption that the population is normally distributed. It often involves the use of matched pairs, here self-adaptive Θ and one of the fixed levels, in which case it tests for a mean difference of zero. Hypothese given in equation 4 is designed to test to compare *ARD* obtained by the replications of self-adaptive Θ for all problem types to that of the three fixed levels, seperately, since we expect *ARD* of self-adaptive Θ, represented by $ARD^{SelfAdaptive}$, is less than $ARD^{Fixed(i)}$, where $ARD^{Fixed(i)}$ is *ARD* value obtained from the fixed level *i* for *i* = 1.0015, 1.0025, 1.0035. Table 2 gives the result of the statistical analysis and p-values which are close to zero indicating $ARD^{SelfAdaptive}$ is statistically different from each $ARD^{Fixed(i)}$ at significant level of .005.

$$H_1 : ARD^{SelfAdaptive} - ARD^{Fixed(i)} < 0 \tag{4}$$

Table 1. Average deviation from the best known using fixed and self-adaptive θ

Application area	Problem Size	Θ 1.0015	1.0025	1.0035	Self-adaptive
VRP	Small	0.0115	0.0063	*0.0012*	**0.0**
	Moderate	0.0102	***0.0047***	0.0253	0.0048
	Large	*0.0162*	0.0434	0.1347	**0.0158**
	Average	0.0126	0.0181	0.0537	**0.0069**
	Std. Dev.	0.0059	0.0191	0.0592	0.0076
PFSP	Small	***0.0095***	0.0276	0.0374	0.0101
	Moderate	*0.0254*	0.0371	0.0427	**0.0016**
	Large	*0.0328*	0.0387	0.0425	**0.0018**
	Average	0.0226	0.0345	0.0409	**0.0045**
	Std. Dev.	0.0101	0.0052	0.0029	0.0051
QAP	Small	*0.0004*	0.0132	0.0293	**0.0004**
	Moderate	*0.0319*	0.0499	0.0563	**0.0013**
	Large	*0.0492*	0.0537	0.0589	**0.0009**
	Average	0.0272	0.0389	0.0482	**0.0008**
	Std. Dev.	0.0205	0.0187	0.0137	0.0007
TDP	Small	0.1795	0.1373	*0.1243*	**0.0129**
	Moderate	0.3068	0.1780	***0.0288***	0.0508
	Large	0.0421	*0.0264*	0.0438	**0.0083**
	Average	0.1761	0.1139	0.0656	**0.0240**
	Std. Dev.	0.2194	0.1503	0.0568	0.0303

Table 2. Results of statistical analysis for comparing of self-adaptive Θ with the fixed levels

Test Hypothesis	Comparison	Mean Difference	p-value
H_1 :	$ARD^{SelfAdaptive} - ARD^{Fixed(1.0015)} < 0$	-.0505	.000[a]
	$ARD^{SelfAdaptive} - ARD^{Fixed(1.0025)} < 0$	-.0423	.000[a]
	$ARD^{SelfAdaptive} - ARD^{Fixed(1.0035)} < 0$	-.0430	.000[a]

[a] Statistically significant different at level of 0.05

3. Implementation

SALS algorithm is compared with some widely used local search based metaheuristics: TS (Glover[16]), SA (Kirkpatrick et al.[17]), and RRT (Dueck[18]). Details of these metaheuristics can be found in the last mentioned references. The aim of this comparative study is to examine the effectiveness and efficiency of SALS relative to the basic versions of TS, SA and RRT metaheuristics on the considered problems, since SALS also is simple algorithm. In this study, TS, SA, and RRT algorithms are coded sticking to the basic principles proposed by the pioneers employing the same neighbor generation mechanism with SALS. Thus, they run under the same base line. Although VRP, PFSP, QAP, and TDP are well-known problems having rather rich and broader literatures, the short descriptions of these problems are given in subsection 3.1, 3.2, 3.3, and 3.4., respectively, to provide a better explanation of neighbor generation mechanism of SALS . Basic structures and acceptance conditions of SALS, TS, SA, and RRT algorithms are defined in subsection 3.6, while neighbor generation mechanisms are introduced in subsection 3.5.

3.1. *Vehicle Routing Problem*

The *Classical* VRP can be described as the problem of designing optimal delivery routes from one depot to a number of customers under the limitations of side constraints to minimize the total traveling cost. Graph theoretic definition of the problem is as follows: Let $G = (V, A)$ be a complete graph, where $V = \{1, ..., n+1\}$ is the vertex set and A is the arc set. Vertices $i = 2, ..., n+1$ correspond to the customers, whereas vertex 1 corresponds to the depot. A nonnegative cost, c_{ij}, associated with each arc $(i, j) \in A$ represents the travel cost between vertexes i and j. Each customer i is associated with a known nonnegative demand, d_i, to be delivered. The total demand assigned to any route may not exceed the vehicle capacity, Q. A fleet of m identical vehicles is located at the depot. Another constraint which is sometimes included in VRP is that the total duration of each route does not exceed a distance limit, L. In the capacity and/or distance constrained VRP, each of the m routes starts and terminates at the depot and each customer is served exactly once by exactly one vehicle. VRP is an NP-hard combinatorial problem and only small-sized problems can be solved optimally. Heuristic methods are commonly used for approximate solutions to VRP in practice.

3.2. *Permutation Flow Shop Scheduling Problem*

PFSP is a production planning problem. There are n jobs to be processed in the same sequence on m machines. Processing time of job i on machine j is given by $t_{ij} \geq 0$. It is assumed that machines can execute at most one job at a time and the operating sequences of the jobs are the same on every machine. The objective is to find the permutation of jobs which will minimize the time between the beginning time of the first job on the first machine and the completion time of the last job on the last machine. PFSP is known to be NP-complete for more than two machines and most of the literature in the last 40 years recommends the heuristic procedures in order to obtain near-optimal solutions to PFSP.

3.3. *Quadratic Assignment Problem*

QAP has remained one of the great challenges in OR. Many practical problems like backboard wiring, facility layout, scheduling, manufacturing and many others can be formulated as QAP. QAP can be described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities to minimize the sum of the product between flows and distances. Mathematically, the problem can be formulated by a flow matrix F whose f_{ij} element represents the flow between facilities i and j and a distance matrix D whose d_{ij} element represents the distance between locations i and j. The goal is the minimization of

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{f}_{ij}\,\mathrm{d}_{x(i)x(j)}$$

over the set of all assignments, where the vector X represents an assignment. QAP is an NP-hard problem. Heuristic methods ranging from simple improvement algorithms to complex metaheuristic algorithms have been proposed for approximate solutions.

3.4. *Topological Design of Computer Networks Problem*

An important stage of the topological design of computer networks is to find the best layout of reliable communication paths among the computers. The problem considered in this study is the backbone network design of computers under the overall network reliability (all-terminal reliability) constraint. Overall network reliability is defined as the probability that every pair of computers can communicate with each other. 2-connectiveness, at least 2 different paths between each pair of nodes, is regarded as a second constraint to increase the reliability of the networks. This topological design problem is NP-hard and has a further complication in that the calculation of overall network reliability is also NP-hard. A backbone network can be modeled by a probabilistic graph G = (N, L, p) where N and L are the set of nodes and edges that correspond to the computers and communication links, respectively, and p is the link reliability. The problem can be modeled as a 0-1 integer programming problem where x_{ij} decision variable takes value 1 if a link exists between nodes i and j, else 0. Thus, the problem is to find the vector, **X**, of the decision variables which minimizes the total cost of the network and satisfies predetermined desired reliability constraint, R_0.

3.5. *Neighbor Generation Mechanisms*

SALS algorithm uses permutation solution representation for VRP, PFSP, and QAP and network solution representation for the TDP. Moving mechanisms to generate neighborhoods for the permutation solution representation of VRP, PFSP, and QAP and the network solution representation of TDP are illustrated in the following subsections.

3.5.1. Permutation Solution Representation

According to the permutation solution representation, a solution point **X** is represented as a vector (x_1, x_2, …, x_D) with dimension D (D = n + m+1 where n is the number of customers and m is the number of vehicles for VRP, D = n where n is the number of jobs for PFSP and n is the number of facilities for QAP). Neighborhood of a solution point **X** is created using five different moving types: Adjacent swap (M_{AS}), general swap (M_{GS}), single insertion (M_{SI}), block insertion (M_{BI}) and reverse location (M_{RL}). These moving types are the most commonly used types of perturbation schemes. Detailed analysis of them can be found in Tian et al.[19] for SA algorithm. Solution representation examples for VRP, PFSP, and QAP are given in Table 3. Definitions and neighborhood sizes of each move type are given in Table 4. Some examples of moving types are also illustrated in Figure 3 for a small (15-customer, 1-depo, 4-vehicle) VRP instance.

3.5.2. Network Solution Representation

Solution **X** is represented using binary coding on a matrix with nxn size. The definitions of the moves are given in Table 5 where n is the number of nodes and $d(i)$ is the degree of node. Figure 4 represents a solution candidate network and its neighbors generated by each moving type.

Table 3. Solution point representation examples for VRP, PFSP, and QAP

Solution example	Explanation
X = [1 12 4 10 7 1 8 9 6 1 5 11 3 2 1]	First vehicle starts its route from the depot 1, then visits customers 12, 4, 10, 7 successively and returns the depot; second vehicle visits customers 8, 9, 6 and third vehicle visits customers 5, 11, 3, 2 successively
X = [3 5 10 15 1 7 8 11 12 14 13 2 4 6 9]	The jobs are processed in the sequence "3 5 10 15 1 7 8 11 12 14 13 2 4 6 9" on each *m* machine
X = [10 1 7 8 3 4 5 12 2 6 9 11]	Facility 10 is assigned to location 1, facility 1 to location 2, facility 7 to location 3 and so on

Table 4. Moving types and neighborhood sizes for permutation representation

Type	Definition	Neighborhood size
M_{AS}	Nodes x_i and x_j are interchanged for $i, j = 1, \ldots, n$ and abs(i-j) = 1.	$N_{AS}(\mathbf{X}) = (n-1)$
M_{GS}	Nodes x_i and x_j are interchanged, for $i, j = 1, \ldots, n$ and abs(i-j)>1.	$N_{GS}(\mathbf{X}) = \dfrac{(n-1)(n-2)}{2}$
M_{SI}	Node x_i is inserted between nodes x_j and x_{j+1}, for $i = 1, \ldots, n$, $j = 1, \ldots, n$-1 and abs(i-j)>1.	$N_{SI}(\mathbf{X}) = (n-1)(n-2)$
M_{BI}	A subsequence of nodes from x_i to x_{i+b} is inserted between nodes x_j and x_{j+1}, for $i = 1, \ldots, n$-1-b, $j = i$+b+1, $\ldots$, n-1 and b = 1, …, n-2.	$N_{BI}(\mathbf{X}) = \begin{cases} \sum_{i=1}^{(n-2)/2} (n-2i)^2, \\ \textit{if} \text{ n is even} \\ \left(\sum_{i=1}^{(n-3)/2} (n-2i)^2 \right) \\ \text{if n is odd,} \end{cases}$
M_{RL}	A subsequence of nodes from x_i to x_j is sequenced in the reverse order for $i, j = 1, \ldots, n$ and abs(i-j)>1.	$N_{RL}(\mathbf{X}) = \dfrac{(n-1)(n-2)}{2}$

Table 5. Moving types and neighborhood sizes for the network solution representation

Type	Definition	Neighborhood size
M_A	Link $x_{i,j}$ takes value 1 for $x_{i,j} = 0$ and $i, j = 1, \ldots, n(n-1)/2$	$\lvert N_{M_A}(\mathbf{X}) \rvert = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1 - x_{ij})$ for $x_{ij} = 0$
M_D	Link $x_{i,j}$ takes value 0 for $x_{i,j} = 1$ and $i, j = 1, \ldots, n(n-1)/2$	$\lvert N_{M_D}(\mathbf{X}) \rvert = \sum_{i=1}^{n-1} d(i)$ for $d(i) > 2$
M_{AD}	Link $x_{i,j}$ takes value 1 and $x_{k,l}$ takes value 0 for $x_{i,j} = 0$, $x_{k,l}$ = 1 and $i, j, k, l = 1, \ldots, n(n-1)/2$	$\lvert N_{M_A}(\mathbf{X}) \rvert \lvert N_{M_D}(\mathbf{X}) \rvert$

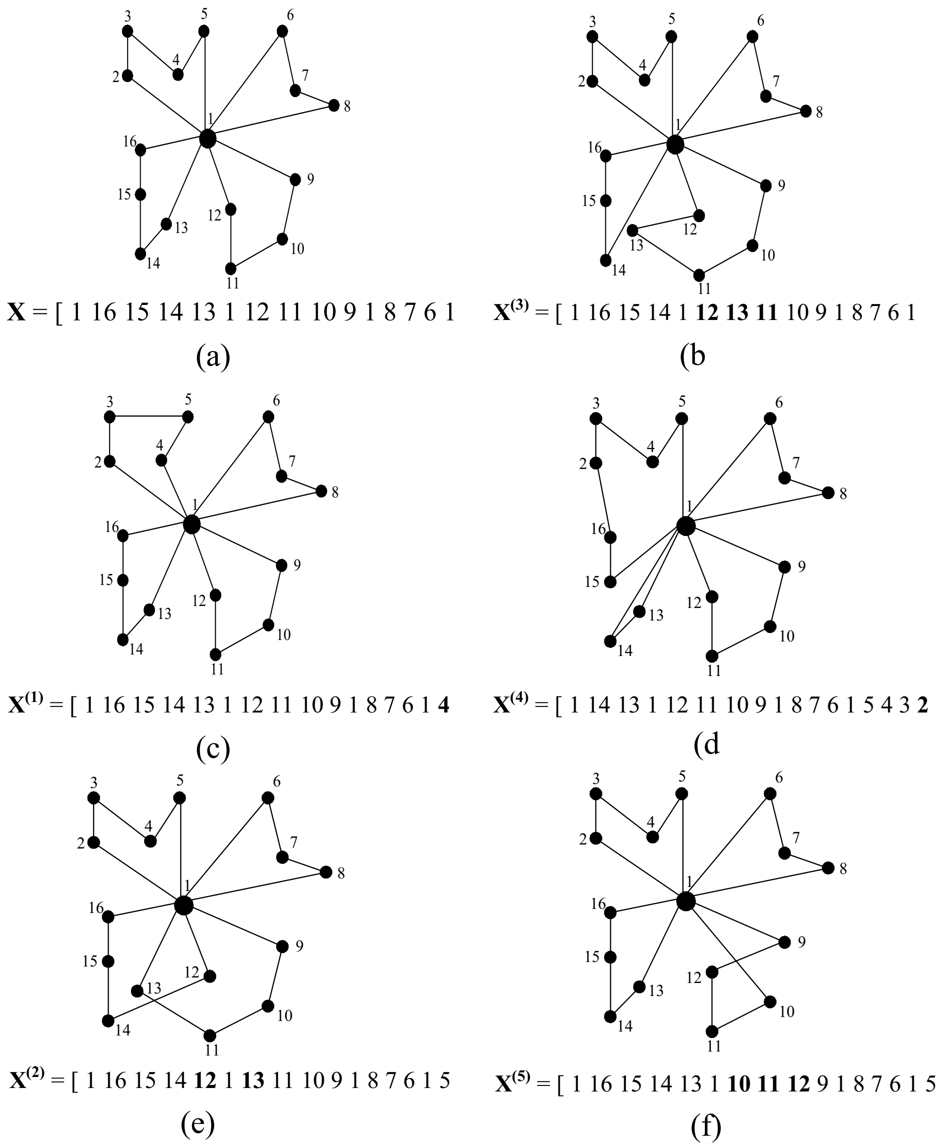
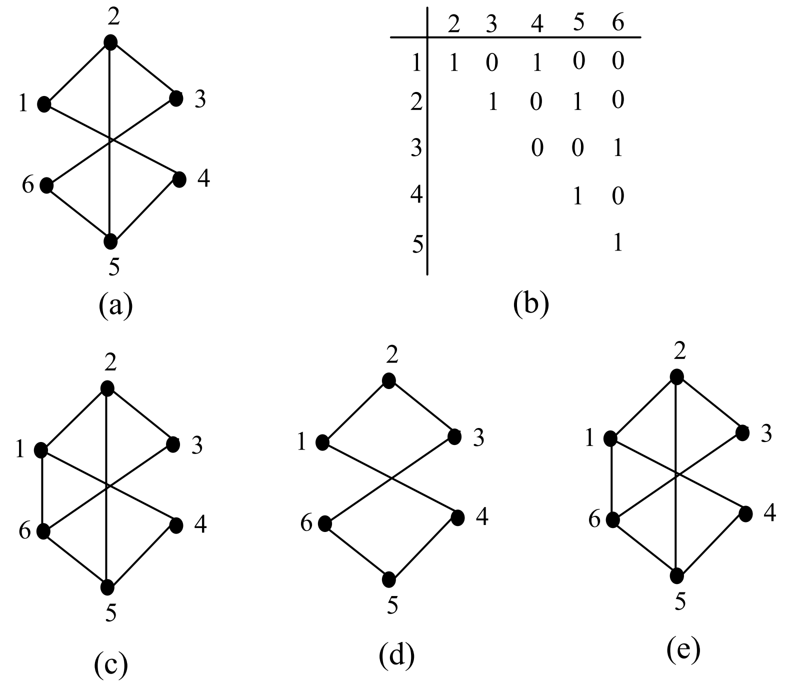


Fig. 3 (a): Current solution **X** (b): $\mathrm{M_{SI}}(\mathbf{X}{:}\ x_5/x_7 - x_8)$ (c): $\mathrm{M_{AS}}(\mathbf{X}{:}\ x_{16}/x_{17})$ (d): $\mathrm{M_{BI}}(\mathbf{X}{:}\ x_2 - x_3/x_{19} - x_{20})$ (e): $\mathrm{M_{GS}}(\mathbf{X}{:}\ x_5/x_7)$ (f): $\mathrm{M_{RL}}(\mathbf{X}{:}\ x_7 - x_9)$



	2	3	4	5	6
1	1	0	1	0	0
2		1	0	1	0
3			0	0	1
4				1	0
5					1

Fig. 4 (a) Current solution **X** (b) Binary coding of **X** (c) $M_A(\mathbf{X}: x_{1,6} = 1)$ (d) $M_D(\mathbf{X}: x_{2,5} = 0)$ (e) $M_{AD}(\mathbf{X}: x_{1,6} = 1, x_{2,5} = 0)$

3.6. *Steps of the Algorithms*

The steps of SALS algorithm are given in Figure 5. At each iteration of the algorithm, a subset $N'(\mathbf{X}: \mathbf{X}^{(s)}, s = 1, \ldots, S)$ is generated from **X** (current solution) by applying S moving types. While in the case of VRP, PFSP, and QAP S is five (explained in Subsection 3.5.1), for TDP S is three (explained in Subsection 3.5.2). The best one, **X'**, among obtained neighbors with best objective value is then selected as a new current solution if it satisfies the acceptance condition "$f(\mathbf{X}') \le \Theta f(\mathbf{X})$", otherwise a new subset $N'(\mathbf{X})$ is generated randomly.

The steps of TS are listed in Figure 6. TS algorithm uses a short-term memory with size *tt*. If a current solution has been created by adjoining p^{th} and r^{th} elements of **X**, moves which disarrange this successive subsequence of the p and r are classified as tabu during next *tt* iterations. At each iteration, the subset $N'(\mathbf{X}: \mathbf{X}^{(s)}, s = 1, \ldots, S)$ is obtained depending on the problem handled and the best solution in the subset which created using a non-tabu move, **X'**, is added to a sampling list, SL, with size *ss*. If the N′ entirely contains tabu moves, then a new N′ is generated until SL is filled with *ss* solutions. However, the aspiration criterion removes the tabu condition when any move yields a better solution than the best solution obtained so far. The best solution, **X''**, in the sampling list is accepted as the new current solution.

SA algorithm is given in Figure 7. The best solution, **X'**, in the $N'(\mathbf{X})$ is recorded as the current solution, if $f(\mathbf{X}') < f(\mathbf{X})$ or $U(0,1) < e^{[f(X)-f(X')]/T}$ is satisfied, where U(0,1) represents a uniformly generated number between 0 and 1. T is a control parameter. The algorithm proceeds by attempting a certain number of neighborhood moves, M, at each temperature, while T is gradually dropped in the ratio of ρ.

Figure 8 represents the steps of RRT algorithm. At each iteration of RRT, the subset $N'(\mathbf{X})$ is generated and the best, **X'**, is then selected as the new current solution if it satisfies the acceptance condition "$f(\mathbf{X}') < f(\mathbf{X}_b^{(i)}) + D$", otherwise a new subset $N'(\mathbf{X})$ is generated randomly.

$i \leftarrow 1$, $C(L^{(i)}) \leftarrow 1$
Randomly create initial solution $\mathbf{X}_0 \leftarrow \mathbf{X}_z$
$\mathbf{X} \leftarrow \mathbf{X}_z$; $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}_z$
Repeat
$\alpha_1 \leftarrow \frac{f(\mathbf{X}_b^{(i)})}{f(\mathbf{X}_z)}$; $\alpha_2 \leftarrow \frac{C(L^{(i)})}{i}$
$\Theta \leftarrow 1 + \alpha 1 \alpha 2$
$i \leftarrow i+1$; $r \leftarrow 0$
Repeat
Select a neighbor solution $\mathbf{X'}$ randomly from the $N'(\mathbf{X})$
$r \leftarrow r + 1$
if $r = |N(\mathbf{X})|$ then $\Theta \leftarrow \Theta + \alpha_1\alpha_2$
Until $f(\mathbf{X'}) \leq \Theta f(\mathbf{X})$
If $f(\mathbf{X'}) < f(\mathbf{X}_b^{(i)})$ then $C(L^{(i)}) \leftarrow C(L^{(i)}) + 1$,
$\mathbf{X}_b^{(i)} \leftarrow \mathbf{X'}$
$\mathbf{X} \leftarrow \mathbf{X'}$
Until $\Theta \rightarrow 1$

Fig. 5 Steps of SALS

Randomly create initial solution, $\mathbf{X}_z$
$\mathbf{X} \leftarrow \mathbf{X}_z$; $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}_z$
Start with empty short-term memory
$i \leftarrow 0$
Repeat
Repeat
Create SL list, $f(\mathbf{X'}_k)$, k = 1,..., *ss*
Select $\mathbf{X''}$ with best $f(\mathbf{X'}_k)$
If $\mathbf{X''}$ is created by nontabu moves or $f(\mathbf{X''}) < f(\mathbf{X}_b^{(i)})$ then $\mathbf{X} \leftarrow \mathbf{X''}$
Otherwise select another $\mathbf{X''}$ from SL list
Until an acceptable solution is found
If $f(\mathbf{X}) < f(\mathbf{X}_b^{(i)})$ then $f(\mathbf{X}_b^{(i)}) \leftarrow f(\mathbf{X})$, $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}$
Update the short term memory
$i \leftarrow i+1$
Until a termination condition is met

Fig. 6 Steps of TS

Randomly create initial solution, $\mathbf{X}_z$
$\mathbf{X} \leftarrow \mathbf{X}_z$; $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}_z$
Select initial temperature, T_b, $T \leftarrow T_b$
Initiated the number of trial neighbors, $m \leftarrow 0$
Repeat
Select a neighbor solution $\mathbf{X'}$ randomly from the $N'(\mathbf{X})$
$m \leftarrow m + 1$
If $f(\mathbf{X'}) < f(\mathbf{X})$ or $U(0,1) < e^{[f(X)-f(X')]/T}$ then $\mathbf{X} \leftarrow \mathbf{X'}$
If $f(\mathbf{X}) < f(\mathbf{X}_b^{(i)})$ then $f(\mathbf{X}_b^{(i)}) \leftarrow f(\mathbf{X})$, $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}$
If $m \geq M$ then $T \leftarrow \rho T$, $m \leftarrow 0$
Until a termination condition is met

Fig. 7 Steps of SA

Randomly create initial solution, $\mathbf{X}_z$
$\mathbf{X} \leftarrow \mathbf{X}_z$; $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}_z$
Select a deviation parameter, *D*
Repeat
Select a neighbor solution $\mathbf{X'}$ randomly from the $N'(\mathbf{X})$
If $f(\mathbf{X'}) < f(\mathbf{X}_b^{(i)}) + D$ then $\mathbf{X} \leftarrow \mathbf{X'}$
If $f(\mathbf{X}) < f(\mathbf{X}_b^{(i)})$ then $f(\mathbf{X}_b^{(i)}) \leftarrow f(\mathbf{X})$, $\mathbf{X}_b^{(i)} \leftarrow \mathbf{X}$
Until a termination condition is met

Fig. 8 Steps of RRT

Table 6. Parameters and selected levels for TS, SA and RRT

Levels	TS		SA			RRT			
						D			
	tt	*ss*	*T*	*M*	ρ	VRP	PFSP	QAP	TDP
-1	$\lfloor\sqrt{n}\rfloor$	n	$0.5n$	$5n$	0.90	5	15	5	20
0	$\lfloor 1.5\sqrt{n}\rfloor$	$2n$	N	$10n$	0.93	10	20	7	30
1	$\lfloor 2\sqrt{n}\rfloor$	$4n$	$2n$	$20n$	0.96	15	25	9	40

4. Computational Study

SALS algorithm is first compared with TS, SA, and RRT algorithms on a suit of selected benchmarking problems and then compared with the other metaheuristics proposed in the related literatures. Since TS, SA, and RRT require an additional process related with parameter tuning, parameter selection studies for these algorithms are given in the next subsection.

4.1. *Parameter Selection*

The basic TS, SA, and RRT algorithms have a set of parameters which shown in Table 6. These parameter sets must be tuned before their run. 3^k factorial experiments are designed individually for this purpose, where k is the number of parameters (k is equal to 2, 3, and 1 for TS, SA, and RRT, respectively). Table 6 also shows the selected parameter levels based on pre-experimentations. While parameter levels of TS and SA are the same for all problem types, the parameter of RRT, *D*, has been changed for each problem type. Twelve separate factorial designs were carried out for each algorithm and each application area. Each algorithm was run 5 times with each parameter combinations and then the analysis of variance was performed at 95% level. Statistical analysis results show that the parameters are statistically significant and solution quality of related algorithm is influenced by parameter levels. Consequently, selected parameter sets which reveal the best solution quality are given in Table 7.

On the other hand, SALS algorithm has a single parameter ϴ and this parameter is tuned adaptively throughout its run as explained previous sections. Significant difference of SALS from other algorithms is that it does not require parameter optimization (tuning) effort.

4.2. *Comparison with TS, SA, and RRT algorithms*

SALS, TS, SA, and RRT algorithms were executed 20 times on a Pentium IV/1000-512 RAM computer. All runs were terminated when the number of solution search reaches pre-determined level. Considered test instances are followed for each problem type:

VRP: 7 instances with 50 - 199 customers (Christofides and Elion[12]).

PFSP: 30 instances of 3 different sizes from the whole benchmark set of Taillard[13]. A sample of 10 instances is provided for each of 50 x 20, 100 x 20, and 200 x 20 (*n x m*) sizes.

QAP: 13 instances with 42 – 100 locations (Skorin-Kapov[14])

TDP: 75 instances of 5 different sizes. A sample of 15 instances is given with known optima for each of 6 – 10 nodes. 3 instances with 15, 20, and 25 nodes with unknown optima are given (Altiparmak[15]).

Performance measures in equation 5-9 were obtained for each algorithm using above defined problem sets separately.

Table 7. Selected parameters for TS, SA and RRT

Application area	TS		SA			RRT
	tt	ss	T	M	ρ	D
VRP	$\lfloor 1.5\sqrt{n} \rfloor$	$4n$	$2.0n$	$5n$	0.93	10
PFSP	$\lfloor 1.5\sqrt{n} \rfloor$	$4n$	$0.5n$	$20n$	0.96	15
QAP	$\lfloor 1.5\sqrt{n} \rfloor$	$4n$	$0.5n$	$5n$	0.90	5
TDP	$\lfloor \sqrt{n} \rfloor$	$2n$	n	$5n$	0.90	20

Relative Deviation percentage:

$$RD_j = 100\frac{O_j^A - O^B}{O^B} \qquad j = 1,\ldots,20 \tag{5}$$

Where, O_j^A is the objective value of considered algorithm obtained from replication j. O^B is reference value (best known or optimum objective value).

Best Relative Deviation: $BRD = \min_j \left(RD_j\right)$ (6)

Average Relative Deviation: $ARD = \frac{\sum_j RD_j}{20}$ (7)

Coefficient of Variation: $CV =$

$$\frac{\sqrt{\sum_{j=1}\left[RD_j - ARD\right]^2 / 20}}{ARD} \tag{8}$$

Average Run Time in Minutes: $ART = \frac{\sum_j Runtime_j}{20}$ (9)

Performance comparisons of the algorithms in terms of defined measures are given in Table 8 for VRP. As shown in this table SALS algorithm outperforms others in terms of *ARD* and *BRD* for all problem sizes. Meanwhile SALS has minimum variability according to *CV*. SA has run time advantage comparing to other algorithms. Similar performance results of SALS are shown in Table 9 for PFSP. SALS is more effective than SA, TS, and RRT algorithms as seen from average results. *CV* of SALS, TS, and RRT are close to each others. TS has the worst effectiveness and efficiency

Table 8. Performance comparison of the algorithms on VRP

Problem	SALS			
customer size	*ARD*	*CV*	*BRD*	*ART*
50	1.42	1.25	0.00	1.1483
75	1.41	0.81	0.00	3.1150
100	0.68	0.29	0.15	3.9508
100	0.00	0.00	0.00	4.1258
120	0.14	0.19	0.00	5.5717
150	1.21	0.57	0.26	8.7982
199	2.87	0.95	1.15	16.3203
Average	1.10	0.06	0.22	6.1472
	TS			
	ARD	*CV*	*BRD*	*ART*
50	2.13	0.0111	0.04	1.1375
75	5.75	0.0066	4.45	2.4258
100	3.85	0.0075	2.46	3.8333
100	1.28	0.0054	0.27	3.7633
120	6.70	0.0287	2.45	5.1725
150	7.18	0.0085	5.85	8.6792
199	9.19	0.0091	6.62	16.3175
Average	5.15	0.0110	3.16	5.9042
	SA			
	ARD	*CV*	*BRD*	*ART*
50	4.87	0.0225	0.00	0.0092
75	9.51	0.0295	3.92	0.0200
100	5.12	0.0148	2.25	0.0317
100	3.65	0.0276	0.48	0.0317
120	9.52	0.0587	2.34	0.0442
150	13.50	0.0209	9.91	0.0683
199	16.96	0.0290	12.69	0.1242
Average	9.02	0.029	4.51	0.0470
	RRT			
	ARD	*CV*	*BRD*	*ART*
50	1.69	0.00928	0.00	1.3133
75	2.44	0.0141	0.832	2.6500
100	1.14	0.00251	0.74	4.3400
100	0.57	0.0015	0.30	4.7707
120	2.72	0.0194	0.22	6.4960
150	2.57	0.0032	2.24	9.7661
199	3.53	0.0076	2.58	16.9201
Average	2.09	0.0082	0.99	6.6080

performance for PFSP while again SA is the fastest algorithm. Table 10 displays the results experienced on QAP. SALS algorithm precisely surpasses other algorithms with respect to average *BRD* and *ARD*. Average *ART* results of SALS and RRT are similar, while the results of TS and SA are better where *ART* reported by SA is the best. For QAP, the worst solution quality performance is belong to RRT algorithm. Finally, Table 11 exhibits performance comparison of the algorithms on TDP. Although, TS has the best *ARD*, the best *BRD* are reported by SALS. SALS algorithm also has shortest *ART* for TDP. RRT algorithm, again, gives the worst solutions to TDP.

The results given in Tables 8-11 are descriptive statistics related with performance metrics of *ARD*, *BRD*, *ART* and *CV* obtained by SALS, TS, SA, and RRT algorithms for all considered problem types. These results especially are encouraging about the solution quality of SALS in terms of *ARD* and *BRD*. A statistical analysis study is also fulfilled to confirm statistically meaningful differences between SALS and other algorithms in terms of effectiveness and efficiency for each problem types. Therefore, the statistical analysis on *ARD*, *BRD* (treated as measures about effectiveness) and *ART* (taken as a measure about efficiency) is performed to test several hypotheses for significance.

Table 9. Performance comparison of the algorithms on PFSP

Problem	SALS			
job x machine	*ARD*	*CV*	*BRD*	*ART*
50 x 20	1.24	0.0029	0.70	11.35
100 x 20	1.48	0.0028	0.95	17.54
200 x 20	1.34	0.0022	0.96	85.04
Average	1.35	0.0026	0.87	37.97
	TS			
	ARD	*CV*	*BRD*	*ART*
50 x 20	3.36	0.0029	2.78	8.03
100 x 20	3.64	0.0021	3.24	31.26
200 x 20	3.37	0.001823	3.02	120.78
Average	3.46	0.0023	3.01	53.36
	SA			
	ARD	*CV*	*BRD*	*ART*
50 x 20	2.19	0.0040	1.51	0.32
100 x 20	2.28	0.0031	1.76	1.58
200 x 20	1.99	0.0025	1.57	6.459
Average	2.15	0.0032	1.61	2.79
	RRT			
	ARD	*CV*	*BRD*	*ART*
50 x 20	1.20	0.0026	0.90	11.79
100 x 20	1.49	0.0022	1.21	18.67
200 x 20	1.43	0.0020	1.16	91.36
Average	1.37	0.0023	1.09	40.61

Table 10. Performance comparison of the algorithms on QAP

Problem	SALS			
location	*ARD*	*CV*	*BRD*	*ART*
42	0.17	0.0014	0.00	5.1702
49	0.15	0.0012	0.07	5.8909
56	0.23	0.0018	0.00	29.4834
64	0.07	0.0005	0.00	39.4208
72	0.18	0.0011	0.00	52.1500
81	0.08	0.0006	0.01	49.6594
90	0.14	0.0009	0.00692	90.3967
100	0.07	0.0002	0.05	210.1782
100	0.06	0.0007	0.02	151.7586
100	0.04	0.0004	0.00406	116.4168
100	0.08	0.0003	0.00	131.9047
100	0.02	0.00014	0.00805	109.3423
100	0.06	0.0004	0.02	177.1752
Average	0.104	0.0007	0.01	89.9190
	TS			
	ARD	*CV*	*BRD*	*ART*
42	0.83	0.0024	0.40	4.8843
49	0.81	0.0020	0.40	7.4601
56	0.98	0.0017	0.64	11.0448
64	0.92	0.0017	0.59	13.8948
72	0.90	0.0015	0.54	22.1710
81	0.71	0.0009	0.50	31.0041
90	0.88	0.0012	0.59	42.1144
100	0.69	0.0003	0.64	206.0432
100	0.71	0.0014	0.37	111.6999
100	0.70	0.0008	0.60	111.8918
100	0.80	0.0010	0.60	103.1167
100	0.81	0.0013	0.52	103.1167
100	0.83	0.0009	0.72	103.0768
Average	0.81	0.0013	0.55	67.0399

Table 10. Performance comparison of the algorithms on QAP (*continued*)

Problem	SA			
location size	*ARD*	*CV*	*BRD*	*ART*
42	0.35	0.0009	0.16	15.1817
49	0.68	0.0011	0.45	16.9134
56	0.79	0.0009	0.69	18.1236
64	0.88	0.0008	0.73	13.9457
72	0.99	0.0008	0.85	22.4009
81	1.01	0.0008	0.82	73.9325
90	1.16	0.0006	1.04	42.0001
100	1.17	0.0006	1.08	102.6517
100	1.09	0.0008	0.97	102.6500
100	1.15	0.0005	1.09	102.6584
100	1.18	0.0006	1.06	102.6698
100	1.09	0.0004	1.03	102.5934
100	1.15	0.0008	0.98	102.6517
Average	0.98	0.0007	0.84	62.9518
	RRT			
	ARD	*CV*	*BRD*	*ART*
42	2.93	0.0043	2.11	4.8202
49	2.15	0.0061	1.07	6.3066
56	2.36	0.0056	1.82	22.4434
64	2.40	0.0038	2.06	15.2400
72	2.57	0.0045	1.84	45.6800
81	2.02	0.0052	1.33	47.9212
90	1.84	0.0036	1.48	68.8599
100	1.52	0.0025	1.12	150.788
100	1.98	0.0044	1.39	149.3803
100	1.99	0.0068	1.00	149.3308
100	1.70	0.0022	1.29	149.3262
100	1.81	0.0039	1.23	149.3800
100	1.42	0.0030	0.99	149.3757
Average	2.05	0.0043	1.44	85.2963

Table 11. Performance comparison of the algorithms on TDP

Problem	SALS			
node size	*ARD*	*CV*	*BRD*	*ART*
6	0.74	0.0131	0.00	1.65
7	1.51	0.0232	0.00	5.69
8	1.21	0.0155	0.05	15.51
9	2.40	0.0329	0.00	28.59
10	2.54	0.0245	0.41	59.10
15	-8.6	0.0371	-13.36	539.33
20	-26.4	0.1251	-38.15	65.132
25	-16.8	0.1190	-29.85	97.82
Average	-5.4	0.049	-10.1	101.60
	TS			
	ARD	*CV*	*BRD*	*ART*
6	0.93	0.0141	0.00	1.11
7	1.54	0.0205	0.00	2.81
8	1.34	0.0138	0.07	11.44
9	3.22	0.0374	0.26	24.27
10	2.61	0.0243	0.55	45.84
15	-0.6	0.0633	-8.14	408.67
20	-32.6	0.0407	-35.93	1203.94
25	-24.3	0.0758	-29.85	1940.58
Average	-6.0	0.036	-9.1	454.83
	SA			
	ARD	*CV*	*BRD*	*ART*
6	5.70	0.0527	1.42	1.91
7	12.90	0.1039	2.04	5.72
8	12.58	0.1105	1.87	12.85
9	12.45	0.1094	1.03	18.92
10	13.87	0.1275	2.87	34.57
15	32.2	0.5363	-12.05	215.71
20	-21.3	0.1687	-34.07	643.67
25	-5.7	0.0246	-8.21	384.79
Average	7.8	0.154	-5.6	164.77
	RRT			
	ARD	*CV*	*BRD*	*ART*
6	7.59	0.0524	2.66	2.59
7	2.74	0.0313	0.20	7.15
8	2.18	0.0220	0.17	22.79
9	3.69	0.0381	0.36	36.92
10	3.52	0.0269	0.68	74.37
15	15.13	0.1643	-10.53	425.13
20	28.74	0.0567	20.74	708.00
25	7.36	0.0393	3.73	1062.00
Average	0.089	0.054	2.3	292.37

The Wilcoxon Signed Rank Test, which is a nonparametric test comparing the pairs, is used in the statistical analysis. The following three alternative hypotheses are defined for each problem type, separately, where *Alg* refers one of the TS, SA or RRT algorithms used in the statistical comparison.

$$H_1^{(1)}: ARD^{SALS} - ARD^{Alg} < 0$$
$$H_1^{(2)}: BRD^{SALS} - BRD^{Alg} < 0 \qquad (10)$$
$$H_1^{(3)}: ART^{SALS} - ART^{Alg} < 0$$

Table 12 gives the Wilcoxon Signed Rank Test results derived from replications of SALS, TS, SA, and RRT algorithms on each test problem of VRP. The table shows that SALS gives smaller *ARD* than TS, SA, and RRT (i.e., the negative mean differences) and these mean differences are all statistically significant with p-values close to zero. The same results are also hold in terms of *BRD*. On the other, ART^{SALS} is greater than that of TS and SA and the mean differences are statistically significant with p-values close to zero. The mean difference between ART^{SALS} and ART^{RRT} is not significant statistically, even ART^{SALS} is smaller than ART^{RRT}.

According to the statistical test on the data gathered from experiments on the test problems of PFSP, Table 13 shows that SALS has smaller *ARD* than TS, SA, and RRT and the mean differences between ARD^{SALS} and ARD^{TS} and between ARD^{SALS} and ARD^{SA} are statistically significant with p-values close to zero while ARD^{SALS} and ARD^{RRT} are statistically similar. On the other hand BRD^{SALS} is all statistically significant smaller than that of TS, SA, and RRT. Additionally, the p-values which are close to zero statistically confirm that run time performance of SALS, ART^{SALS}, is better than ART^{TS} and ART^{RRT} as ART^{SA} is better than ART^{SALS}.

The results of Wilcoxon Signed Rank Test on the data obtained from experiments on the test problems of QAP are shown in Table 14. These results point out that solution quality of SALS in terms of both *ARD* and *BRD* is all statistically better than that of TS, SA, and RRT for QAP. Meanwhile, *ART* of SALS is greater than *ART* of TS, SA, and RRT and the mean differences are statistically significant with p-values smaller than significant level of 0.05.

The last statistically comparison between SALS and the other algorithms is fulfilled for TDP. The results of Wilcoxon Signed Rank Test are given in Table 15. As seen from the table ARD^{TS} is smaller than ARD^{SALS} with p-value of 0.044 which is close to significant level of 0.05 while the mean differences between ARD^{SALS} and ARD^{SA} and ARD^{SALS} and ARD^{RRT} are statistically significant and ARD^{SALS} is better. In terms of *BRD*, SALS and TS are statistically similar for TDP. However SALS is better than both of SA and RRT in terms of *BRD*. Finally, *ART* performance of SALS is better than all other algorithms indicating statistical significance with p-values less than level of 0.05.

Table 12. Results of statistical analysis for SALS, TS, SA, and RRT algorithms on VRP

Test Hypothesis	Comparison	Mean Difference	p-value
$H_1^{(1)}$:	$ARD^{SALS} - ARD^{TS} < 0$	-4.05	.000[a]
	$ARD^{SALS} - ARD^{SA} < 0$	-7.90	.000[a]
	$ARD^{SALS} - ARD^{RRT} < 0$	-.99	.000[a]
$H_1^{(2)}$:	$BRD^{SALS} - BRD^{TS} < 0$	-2.94	.018[a]
	$BRD^{SALS} - BRD^{SA} < 0$	-4.29	.028[a]
	$BRD^{SALS} - BRD^{RRT} < 0$	-.77	.028[a]
$H_1^{(3)}$:	$ART^{SALS} - ART^{TS} < 0$	.243	.000[a]
	$ART^{SALS} - ART^{SA} < 0$	6.10	.000[a]
	$ART^{SALS} - ART^{RRT} < 0$	-.461	.372

[a] Statistically significant different at level of 0.05

Table 13. Results of statistical analysis for SALS, TS, SA, and RRT algorithms on PFSP

Test Hypothesis	Comparison	Mean Difference	p-value
$H_1^{(1)}$:	$ARD^{SALS} - ARD^{TS} < 0$	-2.11	.000[a]
	$ARD^{SALS} - ARD^{SA} < 0$	-.8	.000[a]
	$ARD^{SALS} - ARD^{RRT} < 0$	-.02	.191
$H_1^{(2)}$:	$BRD^{SALS} - BRD^{TS} < 0$	-2.14	.000[a]
	$BRD^{SALS} - BRD^{SA} < 0$	-.74	.000[a]
	$BRD^{SALS} - BRD^{RRT} < 0$	-.22	.000[a]
$H_1^{(3)}$:	$ART^{SALS} - ART^{TS} < 0$	-15.39	.000[a]
	$ART^{SALS} - ART^{SA} < 0$	35.18	.000[a]
	$ART^{SALS} - ART^{RRT} < 0$	-2.64	.000[a]

[a] Statistically significant different at level of 0.05

Table 14. Results of statistical analysis for SALS, TS, SA, and RRT algorithms on QAP

Test Hypothesis	Comparison	Mean Difference	p-value
$H_1^{(1)}$:	$ARD^{SALS} - ARD^{TS} < 0$	-.706	.000[a]
	$ARD^{SALS} - ARD^{SA} < 0$	-.876	.000[a]
	$ARD^{SALS} - ARD^{RRT} < 0$	-1.946	.000[a]
$H_1^{(2)}$:	$BRD^{SALS} - BRD^{TS} < 0$	-.54	.000[a]
	$BRD^{SALS} - BRD^{SA} < 0$	-.83	.001[a]
	$BRD^{SALS} - BRD^{RRT} < 0$	-1.43	.001[a]
$H_1^{(3)}$:	$ART^{SALS} - ART^{TS} < 0$	22.880	.001[a]
	$ART^{SALS} - ART^{SA} < 0$	26.967	.000[a]
	$ART^{SALS} - ART^{RRT} < 0$	4.623	.000[a]

[a] Statistically significant different at level of 0.05

In summary, it is statistically shown that SALS is better than TS, SA, and RRT in terms of both *ARD* and *BRD* for all problem types except TDP in which TS gives smaller *ARD* than SALS. However *BRD* of SALS and TS are not statistically different for TDP. On the other hand SALS has run time advantage respect to TS, SA, and RRT algorithms in terms of *ART* for TDP. For QAP, although, SALS is worse than other algorithms respect to *ART,* it is superior to others in terms of both *ARD* and *BRD*.

Table 15. Results of statistical analysis for SALS, TS, SA, and RRT algorithms on TDP

Test Hypothesis	Comparison	Mean Difference	p-value
$H_1^{(1)}$:	$ARD^{SALS} - ARD^{TS} < 0$	.6	.044[a]
	$ARD^{SALS} - ARD^{SA} < 0$	-13.2	.000[a]
	$ARD^{SALS} - ARD^{RRT} < 0$	-14.3	.000[a]
$H_1^{(2)}$:	$BRD^{SALS} - BRD^{TS} < 0$	-1.00	.087
	$BRD^{SALS} - BRD^{SA} < 0$	-4.550	.000[a]
	$BRD^{SALS} - BRD^{RRT} < 0$	-12.400	.019[a]
$H_1^{(3)}$:	$ART^{SALS} - ART^{TS} < 0$	-353.23	.000[a]
	$ART^{SALS} - ART^{SA} < 0$	-63.17	.001[a]
	$ART^{SALS} - ART^{RRT} < 0$	-191.37	.000[a]

[a] Statistically significant different at level of 0.05

4.3. *Comparison with the Literature*

The aim of the second comparative study is to show the performance of SALS is whether comparable to those of the metaheuristics proposed in the literature. Some of that heuristics used for the comparison are rather sophisticated. These metaheuristics also utilize some problem specific structures and speed-up procedures. On the other hand, SALS has very simple straightforward structure to implement. Therefore, this comparative study takes into account the best solution quality of the heuristics in terms of *BRD*.

4.3.1. Results on VRP

In this application, a feasible initial solution, generated by assigning one vehicle to each customer location, is used to initialize SALS and only feasible neighbors are considered at each iteration of the algorithm. Sizes of benchmark instances by Christofides and Elion[12] have been found insufficient to compare performance of the metaheuristics proposed in the VRP literature. Therefore, 20 larger-sized VRPs by Golden et al.[20] are used to compare the SALS and the metaheuristics listed in Table 16. Table 16 also includes the number of parameters of these metaheuristics and their abbreviations. As outlined in the table, all listed algorithms require parameter tuning process for a number of parameters changing from 1 to 20.

BRD values of the algorithms are shown in Table 17. The reference objective values for each problem to calculate the *BRD*s are given in the *reference value* column of the table. The parameters of all algorithms given in the table, except T-AMP, are tuned for each problem instance separately. T-AMP uses a standard parameter setting for the problem set. As seen in Table 17, SALS has higher solution quality, on average, than five of the metaheuristics. Results of XK-TS, TK-AMP and LGW-RRT algorithms are not reported for the whole problem set since related data is not available in the literature. T-AMP and RDH-AC algorithms which have 9 and 8 parameters, respectively, give similar *BRD* results. Though MB-AGES algorithm gives rather good results for each problem, the parallel implementation of record-to-record algorithm and integer programming, GGW-PRRT$_{IP}$, by Groër et al.[53] outperforms all algorithms. However, large parameter sets of MB-AGES and GGW-PRRT$_{IP}$ make the algorithms complicated to apply different instances.

Table 16. Some successful algorithms for VRP and their parameters

Study	Algorithm	Algorithm Abbreviation	Number of Parameters
Xu and Kelly[21]	Tabu Search	XK-TS	20
Golden et al.[20]	Record-to-Record Travel	GWKC-RRT	3
Tarantilis and Kiranoudis[22]	Adaptive Memory Programming	TK-AMP	7
Tarantilis et al.[8]	Threshold Accepting	TKV-TA1	7
Tarantilis et al.[9]	Threshold Accepting	TKV-TA2	1
Toth and Vigo[23]	Tabu Search	TV-TS	7
Prins[24]	Evolutionary Algorithm	P-EA	7
Reimann et al.[25]	Ant Colony	RDH-AC	9
Tarantilis[26]	Adaptive Memory Programming	T-AMP	8
Li et al.[10]	Record-to-Record Travel	LGW-RRT	5
Mester and Braysy[27]	Active Guided Evolution Strategy	MB-AGES	11
Groër et al.[53]	Parallel Algorithm Combining Record-to-record travel with Integer Programming	GGW-$PRRT_{IP}$	13

4.3.2. *Results on PFSP*

The performance of SALS on the PFSP, described in subsection 4.2, is compared with some of the successful algorithms in the literature. Table 18 shows the considered heuristics with their abbreviations and the number of parameters while Table 19 displays the *BRD* results of these metaheuristics. The studies listed in Table 18 have reported the solution quality results considering different reference objective values. In Table 19, *BRD* values are reported using Taillard's[13] results as reference to overcome this dissimilarity. The results are averaged over the 10 instances of the each size. As seen from the table, SALS outperforms the eight of the algorithms out of twelve in terms of solution quality. NS-MSSA, with 8 parameters, has the best solution quality. Although RS-IG and PTL-DDE have similar *BRD* performances, RS-IG has simplicity advantage from point of parameter tuning. Nevertheless, SALS is the simplest algorithms from the same perspective.

4.3.3. *Results on QAP*

The performance of SALS is compared with the metaheuristics listed in Table 20 on the QAP by Skorin-Kapov[14]. Table 20 shows these heuristics, their abbreviations, and the number of parameters of each algorithm. *BRD* results from the experiments and the results from the QAP literature are displayed in Table 21. As seen from the table, the results of AOT-GGA, LO-HGA, SK-ETS, S-ILS/ES, and JRG-CPTS algorithms are available for the whole problem set. SALS is superior in average to these algorithms, except S-ILS/ES and JRG-CPTS. While CK-TS, T-TS, S-ILS/ES and JRG-CPTS algorithms outperform SALS for the first seven problems, FF-HGA finds the best solutions for the last eight problems.

4.3.4. *Results on TDP*

Effectiveness of SALS for TDP is compared with the algorithms given in Table 22 on the selected test problems, represented with notations L (number of links), p (reliability of the links), and R_0 (overall network reliability requirement), from the whole benchmark set of Altiparmak[15] mentioned in subsection 4.2. As seen in Table 23, SALS, DAS-LGA, and DAB-ACO_{SA} give the optimum results at least one time for all problems. DAS-GA and AA-NN also are able to generate high quality solutions. While DAB-ACO_{SA} has minimum average of *CV*, SALS has lower *CV* than that of DAS-GA and RR-SDA. *CV* results are not applicable for NN approach.

Table 17. *BRD* results for VRP

Pr	n	*Reference value*	XK-TS	GWKC-RRT	TKV-TA1	TKV-TA2	TK-AMP	TV-TS	P-EA	RDH-AC	T-AMP	LGW-RRT	MB-AGES	GGW-$PRRT_{IP}$	SALS
1	241	5623.47	—	3.754	1.070	1.008	0.951	2.004	0.412	0.365	0.951	0.283	0.072	0.000	0.399
2	321	8404.61	—	7.111	1.478	1.285	1.285	1.766	0.515	0.530	0.658	—	0.515	0.362	0.764
3	401	11036.22	—	7.645	1.481	1.397	1.481	3.321	0.000	0.000	0.000	—	0.000	0.000	0.701
4	481	13592.88	—	7.698	0.502	0.838	0.328	9.694	0.233	0.782	0.328	—	0.233	0.233	1.179
5	201	6460.98	—	3.742	0.088	0.000	0.000	3.661	0.000	0.000	0.000	—	0.000	0.000	0.000
6	281	8400.33	—	7.340	0.345	0.326	0.345	6.702	0.148	0.150	0.166	—	0.148	0.150	0.326
7	361	10101.7	—	11.004	1.936	1.708	1.136	4.413	0.929	0.929	1.136	—	0.930	0.929	1.954
8	441	11635.3	—	7.554	2.738	2.867	2.586	3.446	1.663	1.663	2.586	0.526	0.243	0.125	1.980
9	256	579.71	1.620	1.273	2.969	2.698	—	2.353	2.041	1.235	0.987	1.006	0.635	0.000	1.070
10	324	736.26	1.399	1.751	3.908	3.887	—	2.092	2.058	1.971	1.399	0.939	0.720	-0.541	1.373
11	400	912.84	2.173	2.354	3.545	3.533	—	2.542	2.213	1.581	1.132	1.125	0.615	0.056	1.361
12	484	1102.69	3.449	3.128	3.691	3.723	—	4.031	2.820	3.462	2.513	1.227	0.408	0.006	1.293
13	253	857.19	2.786	2.782	1.805	1.724	—	1.354	2.096	0.919	0.912	0.605	0.224	0.000	2.374
14	321	1080.55	3.474	2.142	2.022	2.046	—	1.446	0.527	1.223	0.511	0.287	0.070	0.000	2.071
15	397	1337.92	2.980	1.966	3.447	3.563	—	2.356	2.201	1.517	1.195	1.004	0.546	0.006	1.824
16	481	1612.5	2.739	2.817	4.155	4.016	—	2.469	2.384	1.405	1.379	1.238	0.632	0.072	2.346
17	241	707.76	5.578	1.792	1.469	1.362	—	0.468	0.376	0.141	0.138	0.123	0.004	0.000	0.787
18	301	995.13	7.179	3.425	3.558	3.712	—	2.181	1.977	0.372	1.183	1.279	0.362	0.000	2.008
19	361	1365.6	5.146	2.742	3.150	2.993	—	2.589	0.797	0.117	0.396	0.923	0.092	0.000	0.880
20	421	1818.25	6.420	3.130	2.969	3.004	—	5.367	1.556	0.258	1.068	1.233	0.101	0.000	1.713
Average			3.745	4.258	2.316	2.284	1.014	3.213	1.247	0.931	0.932	0.822	0.328	0.070	1.320

Finally, Table 24 gives the minimum cost results of the heuristics for large-size networks with unknown optimum solutions. It is seen that SALS gives superior results than other metaheuristics for each problem size. For the problem with node size, *N*, is 15, it is seen that obtained solution quality by DAB-ACO_{SA} and SALS is almost same.

Table 18. Some successful algorithms for PFSP

Study	Algorithm	Algorithm Abbreviation	Number of Parameters
Osman and Potts[28]	Simulated Annealing	OP-SA	4
Reeves[29]	Tabu Search	R-TS	2
Reeves[30]	Genetic Algorithm	R-GA	5
Nowicki and Smutnicki[31]	Tabu Search	NS-TS	3
Reeves and Yamada[32]	Genetic Algorithm	RY-GA	5
Grabowski and Pempera[33]	Tabu Search	GP-TS	4
Grabowski and Wodecki[34]	Tabu Search	GW-TS	2
Nowicki and Smutnicki[35]	Modified Scatter Search Algorithm	NS-MSSA	8
Ruiz and Stützle[48]	Iterated Greedy Algorithm	RS-IG	2
Pan et al.[51]	Discrete Differential Evolution Algorithm	PTL-DDE	4
Tseng and Lin[49]	Hybrid Genetic Algorithm and Local Search	TL-GA_{LS}	5
Zobolas et al.[50]	Hybrid Genetic Algorithm and Variable Neighborhood Search	ZTI-GA_{VNS}	3

5. Conclusions

This paper presents a local search algorithm, called SALS, which has a single self-adaptive parameter. SALS algorithm has been tested on four different problem types selected from routing, scheduling, assignment, and topological design areas. SALS algorithm also can be applied to another combinatorial problem if a suitable solution representation scheme, a cost function, and a moving mechanism are described. SALS gathers some feedback information throughout the search to perform a learning process of the parameter θ. The algorithm is successfully applied to VRP, PFSP, QAP, and TDP without any time and talent to manage parameter optimization. From this point of view, transferring of SALS into the real-world applications will be reasonable if the end-users have neither the time nor the experience to fine-tune sophisticated search methods.

Experimental study and statistical analysis show that SALS is the best performing heuristic in terms of solution quality for the mentioned problems comparing the basic TS, SA, and RRT algorithms except topological design problem in which TS is superior to SALS in terms of average solution quality while in the best case SALS and TS have statistically similar performances. As SALS has the shortest average run time for the TDP problems, the run time performance for other problems is obtained as average. Best solution quality results of SALS algorithm also is compared with the performance of heuristics selected from the related literatures. This comparison points out that SALS either competes with these metaheuristics or outperforms the most of them. The proposed algorithm obviously has implementation simplicity and flexibility on different problem types without parameter tuning effort.

Application of SALS to different combinatorial problems is also planned for future directions.

Table 19. *BRD* results on PFSP

nxm	OP-SA	R-TS	R-GA	NS-TS	RY-GA	GP-TS	GW-TS	NS-MSSA	RS-IG	TL-GA_{LS}	ZTI-GA_{VNS}	PTL-DDE	SALS
50x20	2.86	1.55	3.44	-0.36	-0.30	0.17	0.13	-0.90	-0.84	-0.21	0.028	-0.795	-0.22
100x20	2.32	1.07	2.91	-0.66	-0.92	-0.66	-0.68	-1.83	-1.43	-0.33	-0.38	-1.22	-0.78
200x20	1.74	0.08	1.35	-0.80	-0.82	-1.00	-1.12	-1.70	-1.36	-0.43	-0.68	-1.74	-0.89
Average Performance	2.31	0.90	2.57	-0.61	-0.68	-0.50	-0.56	-1.48	-1.21	-0.32	-0.34	-1.25	-0.63

Table 20. Some successful algorithms on QAP

Study	Algorithm	Algorithm Abbreviation	Number of Parameters
Skorin-Kapov[14]	Tabu Search	SK-TS	3
Taillard[36]	Tabu Search	T-TS	4
Skorin-Kapov [37]	Extended Tabu Search	SK-ETS	3
Fleurent and Ferland[38]	Hybrid Genetic Algorithm	FF-HGA	6
Chiang and Kouvelis[39]	Tabu Search	CK-TS	5
Chiang and Chiang[40]	Hybrid Tabu Search	CC-HTS	6
Ahuja et al.[41]	Greedy Genetic Algorithm	AOT-GGA	8
Lim and Omatu[42]	Hybrid Genetic Algorithm	LO-HGA	6
Stützle[43]	Iterated Local Search with Evolution Strategies	S-ILS/ES	6
James et al.[52]	Cooperative Parallel Tabu Search Algorithm	JRG-CPTS	6

Table 21. *BRD* results on QAP

n	*Reference value*	SK-TS	T-TS	SK-ETS	FF-HGA	CK-TS	CC-HTS	AOT-GGA	LO-HGA	S-ILS/ES	JRG-CPTS	SALS
42	15812	0.329	0	0	-	0	0	0	0.354	0	0	0
49	23386	0.641	0	0	-	0	0	0.214	0.188	0	0	0.07
56	34458	0.807	0	0	-	0	0	0.023	0.058	0	0	0
64	48498	1.118	0	0	-	0	0	0.169	0.095	0	0	0
72	66256	0.755	0	0	-	0	0.024	0.272	0.211	0	0	0
81	90998	0.857	0.011	0.011	0	0.011	0.031	0.211	0.123	0	0	0.01
90	115534	0.732	0	0.007	0	0.007	0.095	0.27	0.436	0	0	0.007
100	152002	-	-	0.908	0	-	-	0.191	0.224	0	0	0.05
100	153890	-	-	0.765	0	-	-	0.14	0.296	0	0	0.02
100	147862	-	-	1.219	0	-	-	0.011	0.058	0	0	0.004
100	149576	-	-	0.749	0	-	-	0.17	0.271	0.0013	0	0
100	149150	-	-	0.992	0	-	-	0.231	0.327	0	0	0.008
100	149036	-	-	1.098	0	-	-	0.191	0.411	0.023	0.003	0.02
Average Performance		0.748	0.0016	0.442	0	0.0026	0.0214	0.161	0.235	0.0019	0.000	0.015

Table 22. Successful algorithms for TDP

Study	Algorithm	Algorithm Abbreviation	Number of Parameters
Dengiz et al.[44]	Genetic Algorithm	DAS-GA	3
Dengiz et al.[45]	Genetic Algorithm with Local Search	DAS-LGA	4
Aboelfotoh and Al-Sumait[46]	Neural Network	AA-NN	3
Ramirez-Marquez and Rocco[47]	Probabilistic Solution Discovery Algorithm	RR-SDA	4
Dengiz et al.[54]	Hybrid Ant Colony-Simulated Annealing Algorithm	DAB-ACO_{SA}	8

Table 23. *BRD* and *CV* comparisons on the heuristics for the moderate sized TDP

					DAS-GA		DAS-LGA		AA-NN		RR-SDA		DAB-ACO_{SA}		SALS	
N	*L*	*p*	R_0	*Reference Value*	*BRD*	*CV*	*BRD*	*CV*	*BRD*	*CV*	*BRD*	*CV*	*BRD*	*CV*	*BRD*	*CV*
8	28	0.90	0.90	208	0	0.0211	0	0.0161	0		0	0.0315	0	0.0118	0	0.0171
8	28	0.90	0.95	247	0	0.0183	0	0.0183	0		0	0.0314	0	0.0049	0	0.0132
8	28	0.95	0.95	179	0	0.0228	0	0	0		0	0.0284	0	0	0	0.0151
9	36	0.90	0.90	239	0	0.0497	0	0.0066	0		0	0.0356	0	0.0048	0	0.0152
9	36	0.90	0.95	286	0	0.034	0	0.0325	0.0769	N/A	0	0.0474	0	0.0069	0	0.0295
9	36	0.95	0.95	209	0	0.0839	0	0	0		0	0.0569	0	0	0	0.017
10	45	0.90	0.90	154	0.0128	0.0618	0	0.0223	0		0	0.0791	0	0.0042	0	0.0364
10	45	0.90	0.95	197	0.0496	0.0095	0	0.0177	0		0	0.0448	0	0.0181	0	0.0155
10	45	0.95	0.95	136	0	0.0802	0	0.0185	0		0	0.0618	0	0	0	0.0292
				Average	0.0069	0.0424	0	0.0147	0.0085		0	0.0463	0	0.0056	0	0.0209

Table 24. Best cost comparison on the heuristics for the large sized TDP

N	*L*	*p*	R_0	DAS-GA	DAS-LGA	AA-NN	RR-SDA	DAB-ACO_{SA}	SALS
15	105	0.90	0.95	317		304	268	262	263
20	190	0.95	0.95	926	N/A	270	200	181	167
25	300	0.95	0.90	1606		402	331	322	282

References

1. Blum C and Roli A (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surveys, 35(3):268-308.
2. Glover FW and Kochenberger GA (2003). *Handbook of Metaheuristics*. Series: International Series in Operations Research & Management Science , Vol. 57, Springer.
3. Adenso-Diaz B and Laguna M (2006). Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research, 54(1):99-114.
4. Barr RS, Golden BL, Kelly JP, Resende MGC and Stewart WR (1995). Designing and reporting computational experiments with heuristic methods. Journal of Heuristics, 1(1):9-32.
5. Eiben AE, Hinterding R and Michalewicz Z (1999). Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 3(2):124-141.
6. Battiti R and Tecchioli G (1994). The reactive tabu search. ORSA Journal on Computing, 6(2):126-140.

7. Russell RA and Chiang W-C (2006). Scatter search for the vehicle routing problem with time windows. European Journal of Operational Research, 169: 606-622.
8. Tarantilis CD, Kiranoudis CT and Vassiliadis VS (2002). A backtracking adaptive threshold accepting metaheuristic method for the vehicle routing problem. System Analysis Modelling Simulation (SAMS), 42(5):631-644.
9. Tarantilis CD, Kiranoudis CT and Vassiliadis VS (2002). A list based threshold accepting algorithm for the capacitated vehicle routing problem. Journal of Computer Mathematics, 79(5):537-553.
10. Li F, Golden B and Wasil E (2005). Vey large-scale vehicle routing: new test problems, algorithms, and results. Computers & Operations Research, 32:1165-1179.
11. Osman IH and Wassan N (2002). Reactive tabu search meta-heuristic for the vehicle routing problem with backhauls. Journal of Scheduling, 5(4):263-285.
12. Christofides N and Elion S (1969). An algorithm for the vehicle dispatching problem. *Operational Research Quarterly*, 20:309-318.
13. Taillard, E (1990). Some efficient heuristic methods for the flow shop sequencing problem. European Journal of Operational Research, 47(1), 65-74.
14. Skorin-Kapov J (1990). Tabu search applied to the quadratic assignment problem. ORSA Journal on Computing, 2(1):33-45.
15. Altiparmak, F (1996). Genetik Algoritmalar İle Haberleşme ŞebekelerininTopolojik Optimizasyonu. Phd Thesis, *Gazi University, Turkey*.
16. Glover F (1986). Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 1(3):533-549.
17. Kirkpatrick S, Gelatt Jr. CD and Vecchi MP (1983). Optimization by simulated annealing. Science, 220(4598):671-680.
18. Dueck G (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics, 104:86-92.
19. Tian P, Ma J and Zhang D-M (1999). Application of the simulated annealing algorithm to the combinatorial optimization problem with permutation property: An investigation of generation mechanism. European Journal of Operational Research, 118: 81-94.
20. Golden BL, Wasil EA, Kelly JP and Chao I-M (1998). The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic TG, Laporte G, editors. Fleet Management and Logistics. Boston:Kluwer.
21. Xu J and Kelly JP (1996). A network flow-based tabu search heuristic for the vehicle routing problem. Transportation Science, 30(4):379-393.
22. Tarantilis CD, Kiranoudis CT (2002). BoneRoute: an adaptive memory-based method for effective fleet management. Annals of Operations Research, 115(1):227-241.
23. Toth P and Vigo D (2003). The granular tabu search (and its application to the vehicle routing problem). INFORMS Journal on Computing, 15(4):333-348.
24. Prins C (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 31:1985-2002.
25. Reimann M, Doerner K and Hartl RF (2004). D-Ants: saving based ants divide and conquer the vehicle routing problem. Computers & Operations Research, 31(4): 563-591.
26. Tarantilis CD (2005). Solving the vehicle routing problem with adaptive memory programming methodology. Computers & Operations Research, 32(9):2309-2327.
27. Mester D and Braysy O (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Computers and Operations Research, 34:2964-2975.
28. Osman IH and Potts CN (1989). Simulated annealing for permutation flow-shop scheduling. Omega,17(6):551-557.
29. Reeves CR (1993). Improving the efficiency of tabu search for machine sequencing problems. Journal of the Operational Research Society, 44(4):375-382.
30. Reeves CR (1995). A genetic algorithm for flowshop sequencing. Computers and Operations Research, 22(1):5-13.
31. Nowicki E and Smutnicki C (1996). A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research, 91:160-175.
32. Reeves CR and Yamada T (1998). Genetic algorithms, path relinking and the flowshop sequencing problem. Evolutionary Computation, 6(1):230-234.
33. Grabowski J and Pempera J (2001). New block properties for the permutation flow shop problem with application in TS. Journal of the Operational Research Society, 52:210-220.
34. Grabowski J and Wodecki M (2004). A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Computers & Operations Research, 31:1891-1909.
35. Nowicki E and Smutnicki C (2006). Some aspects of scatter search in the flow-shop problem. European Journal of the Operational Research, 169:654-666.
36. Taillard E (1991). Robust taboo search for the quadratic assignment problem. Parallel Computing, 17:443-455.
37. Skorin-Kapov J (1994). Extensions of a tabu search adaptation to the quadratic assignment problem. Computers and Operations Research, 21(8):855-865.
38. Fleurent C and Ferland JA (1994). Genetic hybrids for the quadratic assignment problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16:173-187.
39. Chiang WC and Kouvelis P (1996). An improved tabu search heuristic for solving facility layout design problems. International Journal of Production Research, 34(9):2565-2585.

40. Chiang WC and Chiang C (1998). Intelligent local search strategies for solving facility layout problems. European Journal of Operational Research,106(2-3):457-488.
41. Ahuja RK, Orlin JB and Tiwari A (2000). A greedy genetic algorithm for the quadratic assignment problem. Computers and Operations Research, 27:917-934.
42. Lim M-H and Omatu S (2002). Extensive testing of a hybrid genetic algorithm for solving quadratic assignment problem. Computational Optimization and Applications, 23:47-64.
43. Stützle T (2006). Iterated local search for the quadratic assignment problem. European Journal of the Operational Research, 174:1519-1539.
44. Dengiz B, Altıparmak F and Smith AE (1997). Efficient optimization of all-terminal reliable networks using an evolutionary approach. IEEE Transactions on Reliability, 46(1):18-26.
45. Dengiz B, Altıparmak F and Smith AE (1997). Local searh genetic algorithm for optimal design of reliable networks. IEEE Transactions on Evolutionary Computation, 1(3):179-188.
46. AboElFotoh HMF and Al-Sumait LS (2001). A neural approach to topological optimization of communication networks, with reliability constraints. IEEE Transactions on Reliability, 50(49):397-408.
47. Ramirez-Marquez, JE and Rocco, C (2008). All-terminal Network Reliability Optimization via Probabilistic Solution Discovery. Reliability Engineering & System Safety, 93(11), 1689-1697.
48. Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research, 177(3), 2033-2049.
49. Tseng, L.-Y., Lin, Y-T., 2009. A hybrid genetic local search algorithm for thepermutation flowshop scheduling problem. European Journal of Operational Research, 198(1), 84-92.
50. Zobolas, G.I., Tarantilis, C.D., Ioannou, G., 2009. Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. Computers & Operations Research, 36, 1249-1267.
51. Pan, Q.-K., Tasgetiren, M.F., Liang, Y.-C., 2008. A discrete differential evolution algorithm for the permutation flowshop problem. Computers&Industrial Engineering, 55, 795-816.
52. James, T., Rego, C., Glover, F., 2009. A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research, 195, 810-826.
53. Groër, C., Golden, B., Wasil, E., 2011. A parallel algorithm for the vehicle routing problems. INFORMS Journal on Computing, 23, 315–30.
54. Dengiz B, Altiparmak F, Belgin O., 2009. Design of reliable communication networks: A hybrid ant colony optimization. IIE Transactions, 42(4), 273-287.
55. Silberholz, J., Golden, B. 2010. Comparison of Metaheuristics. Handbook of Metaheuristics. Eds: M., Genderau and J.-Y. Potvin, Springer.