Makespan minimizing on multiple travel salesman problem with a learning effect of visiting time

CHIARA COLOMBARONI, MOSTAFA MOHAMMADI, GOLMAN RAHMANIFAR
Department of Civil, Constructional and Environmental Engineering (DICEA)
Sapienza University of Rome
Via Eudossiana, 18 - 00184 Rome
ITALY
chiara.colombaroni@uniroma1.it, mohammadi.1755962@studenti.uniroma1.it ,
rahmanifar.1755966@studenti.uniroma1.it

Abstract: - The multiple traveling salesman problem (MTSP) involves the assignment and sequencing procedure simultaneously. The assignment of a set of nodes to each visitors and determining the sequence of visiting of nodes for each visitor. Since specific range of process is needed to be carried out in nodes in commercial environment, several factors associated with routing problem are required to be taken into account. This research considers visitors' skill and category of customers which can affect visiting time of visitors in nodes. With regard to learning-by-doing, visiting time in nodes can be reduced. And different class of customers which are determined based on their potential purchasing of power specifies that required time for nodes can be vary. So, a novel optimization model is presented to formulate MTSP, which attempts to ascertain the optimum routes for salesmen by minimizing the makespan to ensure the balance of workload of visitors. Since this problem is an NP-hard problem, for overcoming the restriction of exact methods for solving practical large-scale instances within acceptable computational times. So, Artificial Immune System (AIS) and the Firefly (FA) metaheuristic algorithm are implemented in this paper and algorithms parameters are calibrated by applying Taguchi technique. The solution methodology is assessed by an array of numerical examples and the overall performances of these metaheuristic methods are evaluated by analyzing their results with the optimum solutions to suggested problems. The results of statistical analysis by considering 95% confidence interval for calculating average relative percentage of deviation (ARPD) reveal that the solutions of proposed AIS algorithm has less variation and Its' confidence interval of closer than to zero with no overlapping with that of FA. Although both proposed meta-heuristics are effective and efficient in solving small-scale problems, in medium and large scales problems, AIS had a better performance in a shorter average time. Finally, the applicability of the suggested pattern is implemented in a case study in a specific company, namely Kalleh.

Key-Words: - Artificial Intelligence, Multiple Traveling Salesman, Learning Effect, Makespan

Received: August 7, 2020. Revised: October 2, 2020. Accepted: October 5, 2020. Published: October 6, 2020.

1 Introduction

Multiple traveling salesman problems (MTSP) lies at the intersection of two primary problems in transportation: uncomplicated TSP and vehicle routing problem (VRP) [1]. This determines that MTSP is an expansion of TSP, in which more than one seller is authorized to exist. On the other hand, MTSP can be analyzed as a relaxed vehicle routing problem, in which there are no constraints on capacity [2]. MTSP defines a variety of routes for salesmen who do not possess any capacity restrictions on the number of goods they can carry.

In the past years, many researchers have investigated MTSP by analyzing numerous constraints such as unique or multiple depots, number of sellers, fixed charges, time windows, bounds on the number of clients per salesman visits, maximizing or minimizing lengths of salesman travels [2]. Also, some investigations such as Svestka and Huckfeldt [3], Bellmore and Hong [4], Russell [5], Hong and Padberg [6]) are pioneer in considering MTSP. Tang et al. [7] presented multiple traveling salesman problem models for hot rolling scheduling in an iron and steel company in China. They introduced a parallel strategy to model the scheduling problem and solve it by utilizing a novel and modified genetic algorithm. Qu et al [8] developed a columnar competitive model of neural networks. This model includes a winner-take-all learning rule to solve MTSP. Yadlapalli et al. [9] addressed MTSP with multi-depot and proposed an approach based on a

Lagrangian algorithm. Ghafurian and Javadian [10] investigated fixed destination multi-depot MTSP. They proposed an Ant Colony Optimization (ACO) algorithm. Their results are compared with accurate solutions achieved by Lingo software. Shim et al. [11] developed a hybrid multi-objective estimation of distribution algorithm with a decomposition framework for multi-objective MTSP. They developed the algorithm's search behavior by hybridizing. This is prepared by local exploration meta-heuristic methods, including hill-climbing, simulated annealing, and evolutionary gradient search. Yuan et al. [12] proposed a new crossover operator called a two-part chromosome crossover for solving MTSP using a Genetic Algorithm (GA). [13] combined genetic algorithm and gravitational force to minimize the traveled distance by MTSP salesman. Soylu [14] considered MTSP with two objective functions separately: one is to decrease the most extended tour length, and the other is to reduce the total length of all tours. The author developed a general variable neighborhood search heuristic and applied it to a real-life problem. This problem exists in traffic signalization networks. Kota and Jarmai [15] submitted mathematical modeling for fixed destination multi-depot MTSP by using an evolutionary programming method. Changer et al. [16] presented a hybrid algorithm based on GA and ACO to solve a solid MTSP in fuzzy rough environments. A solid MTSP is an extension of MTSP where travelers use various transportation means to travel from one port to another.

MTSP objectives include minimizing total distance traveled, the most extended tour length, the total time required, and total tour cost. However, one of the applicable objectives for MTSP is workload balancing [17] Also, these objectives are often designed to ensure salesman satisfaction. Satisfaction factors can be shown by the number of customers visited, the number of delivered goods, tour length, and the required time [18]

There are some studies carried out on MTSP with the objective of workload balancing and salesman satisfaction. Lee and Ueng [19] developed an integer programming model for vehicle routing problems by minimizing the total length and balancing the workload. They developed a novel heuristic algorithm to solve the problem. Rita and Helena [20] exhibited a new multi-objective vehicle routing problem with three objectives and multiple periods. The initial objective is cost minimization, the second is balancing work levels, and the third is about marketing. Lacomme et al. [21] considered the capacitated arc routing problem while minimizing makespan in solving large instances of this NP-hard problem. This is performed via memetic algorithms. Jozefowiezet al. [22] addressed a bi-objective vehicle routing problem to minimize total routes length and to balance the routes. They also applied NSGAII that added two mechanisms for improving NSGAII efficiency.

Lacomme et al. [23] addressed the multi-objective vehicle routing problem with route balancing which minimizes two criteria simultaneously: total routing cost and the discrepancy between highest and lowest route cost. They proposed a multi-start evolutionary algorithm based on two search spaces. Each of them uses a distinct solution presentation. Venkatesh and Singh [24] considered an MTSP with two objectives. The first objective is to minimize the total length traveled by all travelers; the second objective is to minimize the maximum length traveled by any traveler. This objective is about fairness as it tries to balance the workload among salespersons. These researchers recommended two meta-heuristic algorithms, including an artificial bee colony algorithm and an invasive weed optimization algorithm.

Meraj Ajam et al. [25] develops a heuristic that solves a mixed integer program for the post-disaster road clearing problem so as to find the route of a work troop responsible for clearing blocked roads with the aim of minimizing minimize total latency by minimization of makespan. The researchers also develop a metaheuristic based on a combination of Greedy Randomized Adaptive Search Procedure and Variable Neighborhood Search.

In common scheduling problems, job processing times are supposed to be static and autonomous throughout the whole process. Nevertheless, this assumption may not be appropriate due to the phenomenon of learning [26]. The learning effect develops while a function is performed by manual operations and thus, leading to increasing workers' skills [27]. Scheduling problems with learning effect have been an intriguing topic for many researchers in recent years. However, Biskup [28], and Cheng and Wang [29], were the pioneers in presenting scheduling problems with learning effects. Moreover, in recent years, some studies have been accomplished on scheduling with learning effect such as Wang et al. [30], Lee [31], Wu and Wang [32], Salehi Mir and Rezaeian [33], etc. In scheduling problems with learning effects, the job's processing time can be diminished by repeating. The present study considers the learning effect on salesman's skill. Also, customers' visiting time can be minimized by repeating the visit process.

This study attempts to ascertain the optimum routes for salespeople about their skills and the

importance degree of customers. Also, it considers the learning effect on salesman's skills to balance the workload to ensure salesmen's satisfaction. To the best of our knowledge, this problem has not been investigated in the literature. One of the impressive applications of this concept is in the distribution systems of companies.

The framework of this paper is outlined as follows. The problem definition and mathematical modeling are introduced in section 2. Section 3 deals with the proposed meta-heuristic approaches, including AIS and FA algorithm. Parameters tuning is exhibited in section 4. Computational results are implemented in section 5. Ultimately, outcomes are displayed in section 6.

2 Problem definition

This study considers an MTSP in which the customers are classified into various groups, ranging from high-quality customers to low-quality ones according to their importance grade, which is proportional to some measures such as customers' average purchase and diversity of goods which they have bought. It is expected that high-quality customers should buy a significant amount of goods in return to much more time, which allocating an array of them to one salesman can lead to dissatisfaction among salespeople.

Moreover, the distance between customers, which are allocated to a tour, plays a critical role in the salesman's productivity.

As a result of a substantial proportion of traveling time, salespeople must spend most of their time in traveling without meeting customers' demands. Besides, salespeople have distinct levels of skill which influenced by personal qualifications, job experience and training programs which they have gone through and their learning effect to visit the customers which it is determined based on the visiting time, such that the real visiting time is a function of the standard visiting time and customer's location in the tour.

The visiting time of each customer by each salesperson is specified by a salesman's skill and customer's importance degree. Because the completion time of each salesman's tour includes traveling time between customers and costumers visiting time, minimizing makespan leads to salesmen's workload balancing by dispensing customers of each group to salesmen. According to the above explanation, Fig. 1 demonstrates an illustration of this setting. In Fig. 1, per symbol, exhibits a class of customers.

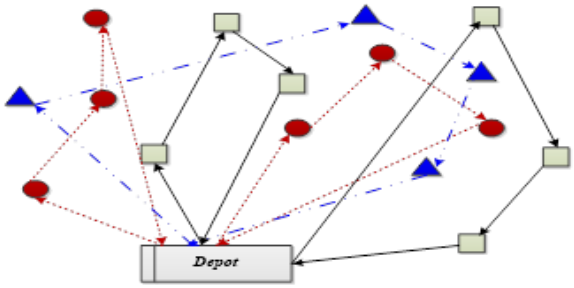

Fig.1. An illustrative pattern of the studied problem

The following assumptions are considered in problem formulation:

- ✓ Each salesman serves just one customer at a time.
- ✓ Each customer will be visited just by one salesman.
- ✓ Traveling time among two customers is deterministic.
- ✓ All salesmen begin their work from the depot.
- ✓ Traveling time is set within depot and customers
- ✓ Visiting time for each customer is varied, which depending on the salesman's skill and customers' importance degree.

Learning effect is determined based on visiting time and by the following equation:

$$pr_{ip} = p_{im} * p^{-\alpha} \quad (1)$$

Where p_{im} is the standard visiting time of customer i by salesman m. pr_{ip} represents a real visiting time of customer i in position p, and α is the learning effect rate. Without loss of generalization, for simplifying the equation, the problem can easily be converted to an unrelated parallel machine scheduling problem with sequence-dependent setup times to minimize the make span. The equivalent information is demonstrated in Table 1.

Table 1. Equivalent vehicles routing problem with parallel scheduling machines

Scheduling problem	routing problem
Machine	Visitor
Job	Customer
Setup time	Traveling time
Processing time	Visiting time

Regarding the above transformation, the model formulation can be presented as follows:

$$\text{Min } Z = \max_i\{C_i\} \quad (2)$$

S.T.

$$\sum_{m=1}^{M}\sum_{p=1}^{P} X_{ipm} = 1 \; i = 1, \dots, N \quad (3)$$

$$\sum_{i=1}^{N} X_{ipm} - \sum_{j=1}^{N} X_{j,p-1,m} \le 0$$
$$m = 1, \dots, M;$$
$$p = 2, \dots, P \quad (4)$$

$$\sum_{i=1}^{N} X_{ipm} \le 1 \; m = 1, \dots, M; \; p = 1, \dots, P \quad (5)$$

$$pr_{im} = \frac{(\sum_{i=1}^{N} p_{im} X_{ipm}) * p^{-a}}{w_i}$$
$$i = 1, \dots, N;$$
$$p = 1, \dots, P;$$
$$m = 1, \dots, M \quad (6)$$

$$C_i - C_j + L(2 - X_{ipm} - X_{j,p-1,m}) \ge pr_{ip} + S_{ij}$$
$$i = 1, \dots, N; i \ne j = 1, \dots, N;$$
$$m = 1, \dots, M;$$
$$p = 2, \dots, P \quad (7)$$

$$C_i \ge (IS_i + pr_{ip}) * X_{i1m} \; m = 1, \dots, M;$$
$$i = 1, \dots, N \quad (8)$$

2.1 Indices

i. j=1.2….. N Index for customers
m=1.2…..M Index for salesmen
p=1.2….. P Index for positions

2.2 Input parameter

N**:** Total number of customers
M**:** Total number of salesmen
$\boldsymbol{P}$**:** Total number of processing positions
p_{im}:The standard processing time (visiting time) for customer$i$ by salesman m.
$\boldsymbol{w_i}$**:** Importance grade of customer i
$\boldsymbol{IS_i}$:Travel time from depot to customer i
$\boldsymbol{a}$**:** The learning effect rate
$\boldsymbol{S_{ij}}$:The travel time from customer$i$ to customer$j$

2.3 Model variables

$\boldsymbol{pr_{ip}}$**:** The actual processing time or visiting time of customer$i$in position p
$\boldsymbol{X_{ipm}}$: If customer i is assigned to position p at tour of salesman m, it is equal to 1 and otherwise is zero
$\boldsymbol{C_i}$**:**Completion time of servicing for customer$i$

2.4 Mathematical model

Relation (2) minimizes the maximum completion time, so the difference between the greatest and smallest workload assigned to visitors are balanced. Constraint (3) by expanding inner summation which over the all position in a salesman's tour and then continuing by outer summation over salesman, ensures that each customer is visited by one of the current positions on the only one salesman tour. Constraint (4) ensures that until one position on a salesman's tour is empty, customers are not assigned to subsequent positions. So, if the costumer i is allocated to position p at tour of salesman m, then so as to satisfy inequality condition the previous position j on that tour must be equal 1.

Otherwise the position j can be zero or one. Constraint (5) represents that the summation over customers for each position of each salesman's tour can be at most one which guarantees that for current position, at most, one node can be assigned. Constraint (6) measures real processing time whenever customer i is assigned to the position of salesman m. The real processing time is affected by the position of customer in the salesman tour and the grade of customer. Constraint (7) guarantees that the completion time of customer service in the sequence of a salesman's tour, is at least equal to the sum of completion times of the preceding customer's service time, traveling time between customers and visiting time of the present customer. L is assumed to be a large number to satisfy the inequality condition in a case that customer I and j are not two consecutive customers in a tour. Constraint (8) measures the completion time of servicing the customer for each salesman by considering the traveling distance from depot to the first customer in the tour. Constraints (9) defines types of decision variables.

3 Solution approach

MTSP is a generalization of the well-known traveling salesman problem [34] which is an NP-hard problem [35], MTSP is more complicated than TSP because it involves finding a set of Hamilton circuits without sub-tours for m>1 salesman to serve a set of n>m customers, such that per customer is visited by precisely one salesman. As TSP belongs to the class of NP-hard problems [5], it is apparent that MTSP is an NP-hard problem, too. This indicates that MTSP computational time grows exponentially by the increase of distribution points. Thus exact algorithms are not intelligent in solving MTSP with large dimensions. Accordingly, it is reasonable to concentrate on developing some practical approximation algorithms instead of the

exact method. In the following subsections, two meta-heuristic algorithms including Artificial Immune System (AIS) and Firefly Algorithm (FA) are elaborated for MTSP.

3.1.Artificial immune system

AIS is a stochastic computational technique based on a biological immune system [36]. Biological Immune System (BIS) is a very complicated, robust, and adaptive system with specific functional components protect it against infectious diseases caused by viruses, bacteria, etc.

For carrying out protection, distinguishing between the body's own cells as the self-cells and foreign pathogens as the non-self-cells or antigens must be done by immune system. Then, the immune system eliminates non-self-cell or antigen by an immune response. With the help of categorizing antigens, a suitable defensive mechanism is activated to enable immune system to develops a memory to enable more efficient responses in case of further infection by the similar antigen. Clonal selection theory explains how the immune system fights against an antigen. According to this Clonal selection, only the cells which recognized as antigen are selected for proliferation.

The selected cells are affected by an affinity maturation process that enhances their affinity to the selected antigens. Clonal selection operates on B-lymphocytes and B cells created by the bone marrow and also on T-lymphocytes or T cells produced by the thymus. By the exposure of the body to an antigen, B cells respond by secreting certain antibodies.

Thereafter, a signal from the T-helper cells, a subclass of T cells, stimulates the B cell to proliferate and mature into terminal antibody-secreting cells called plasma cells. The affinity level determines the proliferation rate, i.e. higher the affinity level of B cells is, more clones will be generated. This process of selection and mutation in B cells is named affinity maturation [37].

Inspired by this natural immune system of human body, an optimization algorithm named as artificial immune system optimization (AIS) has been developed. AIS, similar to the genetic algorithm (GA), is a population-based evolutionary algorithm. It starts with a population of antibodies. Each of them represents a point in the solution space. Each antibody contains an affinity value about its solution quality. Such that, the solution (antibody) with the better objective value has a higher affinity value. Some antibodies with higher affinity values are selected and proliferated. Afterward, the clones are mutated, and a matured population of antibodies is generated. Eventually, the worst antibodies in the initial population will be replaced by better-mutated antibodies.

3.1.1. Antibody representation and initial population

The first step in the proposed AIS is to make an adequate antibody representation. Suppose that there are n jobs to be assigned to m machines.

An antibody is represented by a string of n+m-1 distinct parts, composed of n job parts (numbered from 1 to n) and m – 1 partitioning parts (numbered from n+1 to n+m-1). Such that, the antibody should be decomposed into m sub-antibodies.

Each sub-antibody represents the sequence of jobs on a machine. An example of 8 jobs and three machines is demonstrated in Figure 2. In this example, jobs 2, 1, and five are assigned to machine 1; jobs 4, 7and 3 are assigned to machine 2; jobs 8 and 6 are assigned to machine 3.

As mentioned before, m-1 genes I, including genes numbers 9, 10 a, reapplied to separate the salesmen. The initial population comprises pop-size antibodies. Each antibody is generated in a stochastic procedure from the solution space.

2	**1**	**5**	**9**	**4**	**7**	**3**	**10**	**8**	**6**

Fig.2. A typical solution representation

3.1.2. Affinity calculation

Each antibody has an affinity value, which is calculated based on an affinity function. The affinity relation used in this study is as follows:

$$\text{Affinity (z)} = \frac{1}{\text{Makespan(z)}} \quad (11)$$

Where z is considered as an antibody, from the above equation, it can be observed that the lower the objective value, the higher the affinity value.

3.1.3. Cloning

In this study, the cloning procedure is based on a process that has been implemented successfully by many researchers [36] [38].

During the search procedures of the proposed AIS, n_c antibodies with the highest affinity function values are elected from the antibody population to

design a new set of high-affinity antibodies. These antibodies are selected for cloning. The clone number of each selected antibody is calculated as $(n_c - k + 1)$, where k denotes the antibody with the k the highest affinity value in antibody population. Thus, the whole number of clones generated according to the above equation is $n_c(n_c + 1)/2$. It can be mentioned that antibodies with higher affinity values have more clones.

3.1.4. Mutation

In this step, the matured population is produced by mutating the antibodies of the clone population. In this research, a mutation approach based on the destruction and construction phases of the iterated greedy approach is applied [40]. The proposed destruction and construction mutation mechanism is accomplished through the following steps:

(Destruction phase) Consider a sequence of antibody and randomly pick three unrepeatable components of it. Then exclude them, respectively. As a result, a partial solution will be achieved.

(Construction phase) Again, randomly select three different positions in the obtained partial solution and sequentially insert removed components into the chosen positions.

Mutation in AIS is implemented on the antibodies of the clone population according to their affinity values. Therefore, each antibody may be mutated for more than one time. In this research, several mutation loops for each antibody are equal to the ranking number of its initial antibody (k). For instance, an antibody with the highest affinity value is mutated only one time.

3.1.5. Antibody population update

In each generation of the proposed AIS, a new population is generated. This is performed by combining the best antibodies of the current population and the best antibodies of a matured population. Therefore, the worst n_c of the current population is replaced with the best n_c of the mutated antibodies. Fig 3. Represents an example of the proposed mutation operator.

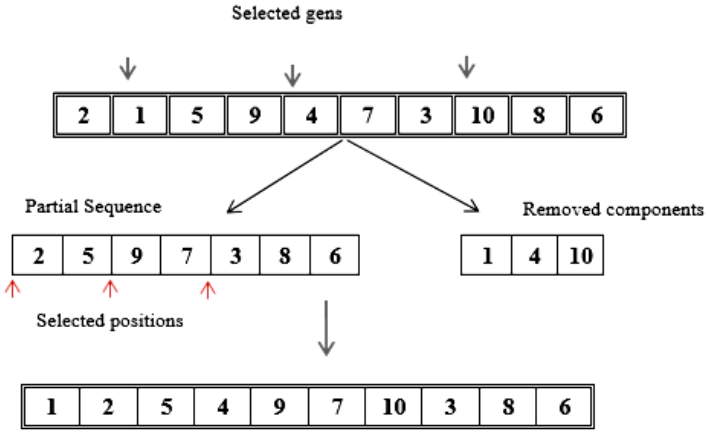

Fig 3. An example of mutation operator

3.2.Firefly Algorithm

Firefly Algorithm (FA) is one of the well-known swarm intelligence methods which was presented by Yang [41].

FA is an accidental and nature-inspired meta-heuristic algorithm that can be implemented for solving the hardest optimization problems. Swarm intelligence belongs to an artificial intelligence discipline that became increasingly popular over the last decade.

It is inspired by the collective behavior of a social swarm of ants, termites, bees, worms, a flock of birds, and classes of fish. Although swarms consist of relatively unsophisticated individuals, they could present a coordinated behavior that leads the swarm to their desired goals. Formulating FA is possible because the flashing light can be formulated in such a way that it is correlated with the objective function. The firefly algorithm was extended by idealizing some of the flashing characteristics of fireflies.

FA utilizes the following three fundamental rules:

-All fireflies are unisexual, which indicates that they attract each other regardless of their sex.

-The degree of attractiveness of a firefly is proportional to its brightness, thus for any two flashing fireflies, the less-brighter one will move toward the brighter one, and their attractiveness will decline while their distance increases. If there is no brighter one than a unique firefly, it will move haphazardly.

-The brightness or light intensity of a firefly is designated based on the objective function. For the maximization problem, the brightness can be proportional to the objective function.

3.2.1. Structure of the Firefly Algorithm

This section is adopted from Fister et al.[40], which has presented a comprehensive review of FAs. In order to design FA properly, two significant variables should be defined: light intensity and attractiveness. Light intensity is based on the distance from the perceiver's eyes. Light intensity I of a firefly representing the solution s is proportional to the value of fitness function ($I(s) \propto f(s)$), while light intensity I(r) varies according to the following equation:

$$I(r) = I_0 e^{-\gamma r^2} \quad (12)$$

I=Light intensity
I_0=Light intensity at an initial or original light intensity
γ=Light absorption coefficient
r=Distance between fireflies I and j

Attractiveness is proportional to brightness. For any two fireflies, the less bright one will be attracted to the brighter one; however, intensity decrease as their mutual distance increases.
Attractiveness (β) can be defined according to the following equation:

$$\beta = \beta_0 e^{-\gamma\, r2} \quad (13)$$

β_0= Attractiveness at r is 0

While the intensity is a definitive measure of emitted light by the firefly, attractiveness is a relative measure of the light that should be seen by a perceiver. This means that attractiveness is judged by other fireflies [41].
The following equation can define the distance between the two fireflies:

$$r_{ij} = |x_i - x_j| = \sqrt{\sum_{k=1}^{k=d} (x_{ik} - x_{jk})^2} \quad (14)$$

If a firefly is attracted toward the more attractive firefly j, its movement is defined as follows:

$$x_i = x_i + \beta_0 e^{-\gamma r^2_{ij}} (x_j - x_i) + \alpha \varepsilon_i \quad (15)$$

In equation (15), parameter γ has a substantial effect on the convergence rate. Although the value of this parameter can theoretically be any value of interval $\gamma \in [0\ \infty)$, its setting depends on the problem.

In brief, FA is controlled by three parameters which can be defined in the following:

α= Randomization parameter
β= Attractiveness
γ= Absorption coefficient

According to the parameter setting, firefly distinguishes two asymptotic behaviours. The former appears when y approaches to zero (y→0). The latter is when y is nearing infinity or is too large (y→∞). If y approaches to zero, the attractiveness becomes β=β_0. That is, attractiveness is constant anywhere within the search space. If y is nearing infinity or is too large (y→∞), the second term emerges from Eq. (15), and the firefly movement becomes a random walk. In fact, any implementation of a firefly is between these two asymptotic behaviours.

As a result, the basic FA can be formulated as follows:

Function firefly

Objective function f(x), x= ($x_1 \ldots x_d$)
Generate an initial population of fireflies
Evaluate light intensity Ii for firefly at x_i by using f (x_i) for all fireflies
Iteration=I;
While (Iteration<=max_iteration)
{
For (i=I; I<=population _ size; i++)
{
For (J=I; j<=population _ size; j++)
{
If (Ii>Ii) Move firefly i towards j in d-dimension;
Attractiveness varies with distance r via exp [- r];
Evaluate light intensity of new solutions;
}
{
Update current best solution;
Iteration++;
}
Return current best solution;
End Function
Function Decoding
A permutation of n different nodes of the graph is held in array F
For (i=1; i<=n; i++)
{
If (F[i]>=i)
Form a new partition and assign node I to it.
Else
Assign me to the same partition as node F[i]'s partition.
}
End Decoding

3.3.Implementation of FA for MTSP

The solution representation of the firefly algorithm for MTSP is the same as the AIS algorithm. Originally, FA is designed to solve continuous optimization problems [42] [43] However, FA can also be discretized to solve permutation problems. In discrete problems, such as combinatorial, binary, and

categorical ones, it is inevitable to decrease the number of possible states of feasible solutions. This is performed by discretizing a continuous space by transforming continuous values into a restricted number of possible states. There are numerous discretization methods. One of them is a random key (RK). RK encoding scheme can be used to transform a position from a continuous space into an integer/combinatorial space [44] [45]. Suppose that there are n customers to be assigned to m salesmen. RK is created by generating n+m-1 random numbers from [0, 1). To decode RK, numbers are sorted in ascending order with candidate indices. For example, the continuous solution vector $\bar{x}$= (0.90,0.20,0.30,0.35,0.5) can be decoded as $\bar{x}$= (5,2,1,3,4). The graphical form of this example is shown in Fig. 4.

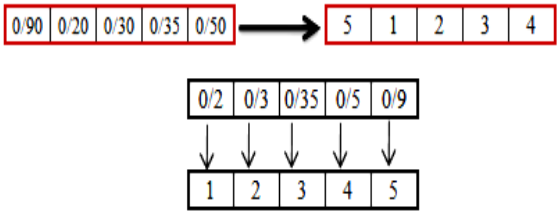

Fig.4. Graphical representation of random key encoding

3.4.Parameters Tuning

It is apparent that the quality of solutions received from per algorithm depends on the various levels of parameters. Therefore, it is imperative to apply the Design of Experiment (DOE) approach to investigate the effect of different levels of parameters on the performance of algorithms. To tune the parameters, the Taguchi method [46] is implemented in the DOE approach. This method uses a specific set of arrays called orthogonal arrays. These standard arrays generate a small number of experiments that contain full information of all factors that affect the performance of algorithms [47].

The initial step of the Taguchi method is to distinguish the significant parameters of algorithms and their levels. AIS controllable parameters include a maximum number of iterations(Maxit), a number of members in population (popsize)and the amount of high-affinity antibodies to be selected for cloning procedure (n_c). FA controllable parameters consist of Combination of a maximum number of iterations and number of population (Maxit, popsize), the light absorption coefficient (γ), attractiveness coefficient (β_0), and randomization parameter (α). Different levels of these parameters are shown in Tables 2 and. Table 3. AIS's parameters and their levels.

Table 2. FA's parameters

Parameters	Level 1	Level 2	Level 3	Level 4
$Maxit$	200	300	400	500
$popsize$	70	80	95	110
n_c	15	20	25	30

Table 3. FA's levels

parameters	Level 1	Level 2	Level 3	Level 4
$(Maxit.popsize)$	(100,40)	(80,50)	(50,80)	(40,100)
γ	0.1	1	5	10
β_0	0.05	0.1	0.5	1
α	0.1	0.5	0.9	0.99

Since AIS has three factors with four levels and FA has four factors with four levels, the proper orthogonal array for both algorithms is L16. Therefore, per combination of parameters employed five instances of various sizes. After running all experiments, the Relative Percentage Deviation (RPD) method is applied to normalize the acquired results. In doing so, the average value of results is reported for each parameter combination as a response variable. RPD is calculated for per algorithm by the following equation:

$$\mathrm{RPD} = \frac{|\mathrm{sol} - \mathrm{Best}|}{\mathrm{Best}} \times 100 \qquad (16)$$

Where sol the solution is acquired by each experiment, and Best is the best value of the solution obtained from overall combination test problems. Finally, to determine the optimal level of the parameters for each algorithm, two measures are employed in Taguchi analysis: average (RPD) as the response variable and signal-to-noise (S/N) ratio as a control variable. Figs. 5 and 6 display average S/N ratios for AIS and FA algorithms, respectively. As illustrated in Fig. 5, better robustness of AIS is achieved. In this case, the factors are set as follows: Max it=500, pop size=80, n_c=30. Based on Fig. 6, suitable parameters for FA are (Max it,popsize)= (40,100), γ=0.1, β_0=0.05, α=0.1.

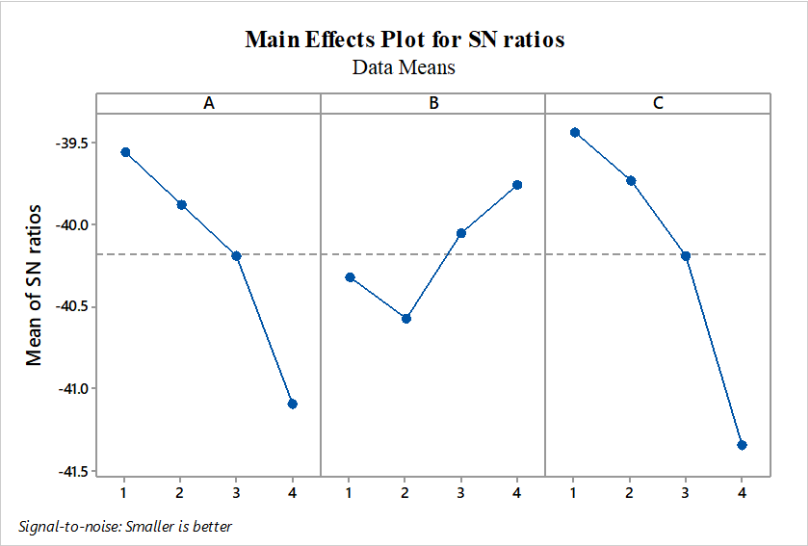

Fig 5. S/N ratio plot for AIS

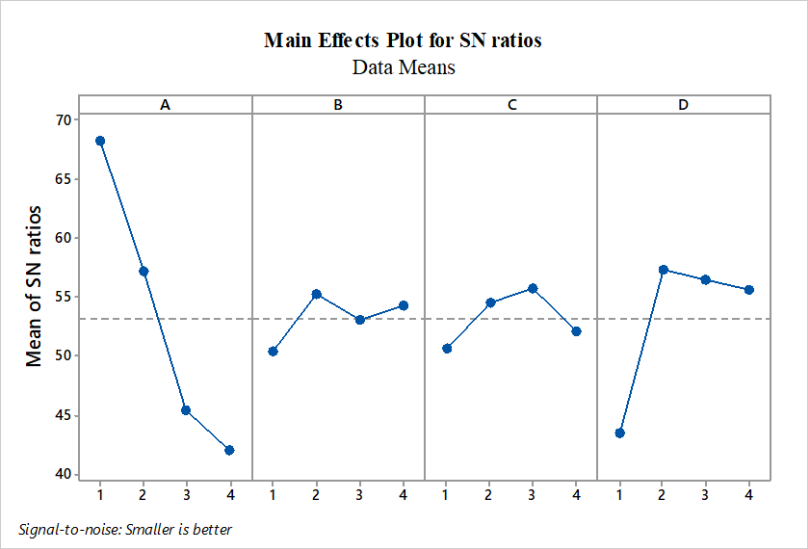

Fig 6. S/N ratio plot for FA

4 Experimental design and analysis

In order to evaluate the performances of the suggested mathematical model and meta-heuristic algorithms in this study, some computational experiments are applied by applying randomly generated test problems. Experimental tests are classified into three small, medium, and large groups. 30, 20, and 20 random examples have been produced, for small, medium, and large groups, respectively. Detailed information about the number of salesmen and customers for each group are presented in Table 4. As shown in Table 4, seventy test problems have been named from Num01 to Num70.

Visiting times, traveling times, salesman's skill, and the quality grade of each customer are generated for each test problem as follows:

Visiting times:
$p_{im} \rightarrow \mathrm{U}[10\ 20]$
Traveling time:
S * average of p_{im}
S = $(0.5 - 1 - 1.5)$
Salesman's skill → $(0.8 - 0.9 - 1 - 1.1 - 1.2)$
Quality grade of each customer → $(1.1 - 1.5 - 1.3)$
Learning rate → $(0.05.\ 0\ .1\ 0.15)$

Table 4. Detailed information about test problems

Problem Size	Number of Customers	Number of Salesmen					
		2	3	4	5	7	8
Small	5	Num01	Num06	-	-	-	-
		Num02	Num07	-	-	-	-
		Num03	Num08	-	-	-	-
		Num04	Num09	-	-	-	-
		Num05	Num10	-	-	-	-
	6	Num11	Num16	-	-	-	-
		Num12	Num17	-	-	-	-
		Num13	Num18	-	-	-	-
		Num14	Num19	-	-	-	-
		Num15	Num20	-	-	-	-
	8	Num21	Num26	-	-	-	-
		Num22	Num27	-	-	-	-
		Num23	Num28	-	-	-	-
		Num24	Num29	-	-	-	-
		Num25	Num30	-	-	-	-
Medium	40	-	-	Num31	Num36	-	-
		-	-	Num32	Num37	-	-
		-	-	Num33	Num38	-	-
		-	-	Num34	Num39	-	-
		-	-	Num35	Num40	-	-
	50	-	-	Num41	Num46	-	-
		-	-	Num42	Num47	-	-
		-	-	Num43	Num48	-	-
		-	-	Num44	Num49	-	-
		-	-	Num45	Num50	-	-
Large	70	-	-	-	-	Num51	Num56
		-	-	-	-	Num52	Num57
		-	-	-	-	Num53	Num58
		-	-	-	-	Num54	Num59
		-	-	-	-	Num55	Num60
	100	-	-	-	-	Num61	Num66
		-	-	-	-	Num62	Num67
		-	-	-	-	Num63	Num68
		-	-	-	-	Num64	Num69
				-	-	Num65	Num70

4.1.Computational Result

Some experiments have been preliminarily performed on small instances to evaluate the quality of a proposed mathematical model and to benchmark the results based on the recommended meta-heuristics. This is performed in terms of solution quality and CPU times. Accordingly, all 30 small test problems are optimally solved using Lingo 9.0 software with a branch-and-bound (B&B) approach. Our runtime was limited to 3 hours. However, it is impossible to obtain an optimal solution for some of the small problems in 3 hours of central processing unit (CPU) time. Therefore, the best solution obtained after 3 hours is reported for these small problems. All small problems are also solved by the proposed AIS and FA algorithms. Each algorithm is run seven times for each problem, and the best one is selected as the final solution. The proposed algorithms were coded in MATLAB R2012. All test problems were solved on a PC Pentium IV, 2.4 GHz speed with 6 GB of RAM. Computational results for small problems are shown in Table 5.

Table 5 exhibits optimal solutions and CPU times obtained from Lingo 9.0 software. Also, the best and the average results and average CPU times achieved from algorithms in 7 replications for small size problems are displayed in this table too. From Table 5, it can be observed that the proposed AIS and FA are responsible for gaining exact solutions for smallest size problems in reasonable CPU times. Only for problems 6, 16, 27, and 30, the algorithms could not detect an exact solution. Consequently, it can be stated that the proposed meta-heuristics are robust enough for solving the small-sized problems. The fig.7 indicates exponential behaviors of CPU times in the exact method against the polynomial behavior of the proposed meta-heuristics in case of developing the problem size.

As can be observed in Fig. 7, the exact method spends longer CPU times drastically for achieving an optimal solution in small problems. In Fig. 7, the horizontal axis indicates 30 small problems from Num 01 to Num30.

Table 5. Computational results for small-size problems

Problem Name	LINGO		AIS			FA		
	Best	CPU times	Average	Best	CPU times	Average	Best	CPU times
Num01	74.9	54	75.86	74.9	9.902	75.52	74.9	39.52
Num02	74.9	62	76.21	74.9	9.667	77.7	74.9	39.34
Num03	54.2	51	56.09	54.2	11.2	56.53	54.2	39.17
Num04	52.3	58	53.23	52.3	12.04	54.41	52.3	39.11
Num05	68.3	62	69.34	68.3	10.2	73.13	68.3	40.13
Num06	39.8	135	42.93	41.6	9.314	45.39	41.6	37.61
Num07	41	228	42.89	41	9.362	43.05	41	37.36
Num08	39.8	266	39.84	39.8	10.31	41.77	39.8	37.85
Num09	58.1	255	58.39	58.1	8.891	61.9	58.1	38.95
Num10	34.4	242	35.33	34.4	9.06	35.65	34.4	38.11
Num11	70.3	310	71.69	70.3	9.135	73.94	70.3	38.01
Num12	50.9	242	55.39	50.9	8.937	59.37	50.9	38.27
Num13	60.9	260	63.67	60.9	9.015	64.23	60.9	37.68
Num14	50.3	237	51.14	50.3	9.964	54.13	50.3	40.62
Num15	78.3	239	80.04	78.3	10.07	81.26	78.3	39.57
Num16	44.7	264	45.23	46.1	10.35	47.92	46.1	39.86
Num17	59.1	327	59.11	59.1	9.958	61.27	59.1	40.6
Num18	42.6	342	45.52	42.6	9.734	47.39	42.6	40.91
Num19	53.2	279	54.56	53.2	9.863	62.31	53.2	39.58
Num20	38.7	251	39.86	38.7	9.603	40.33	38.7	38.85
Num21	61.9	10800	63.13	61.9	9.706	67.73	61.9	39
Num22	49.8	10800	51.7	49.8	9.8	54.02	49.8	40.96
Num23	64.5	10800	67.06	64.5	9.835	69.9	64.5	39.86
Num24	62.8	10800	63.4	62.8	9.791	68.73	62.8	39.31
Num25	85	10800	86.94	85	9.881	92.18	85	39.2
Num26	62.2	10800	63.06	62.2	10.21	64.77	62.2	40.98
Num27	50.3	10800	51.2	54.1	10.27	56.04	54.1	39
Num28	54.1	10800	59.12	54.1	10.2	65.64	54.1	42.62
Num29	61.5	10800	62.72	61.5	10.11	68.17	61.5	39.31
Num30	66	10800	70.21	77	10.03	77.82	77	37.85

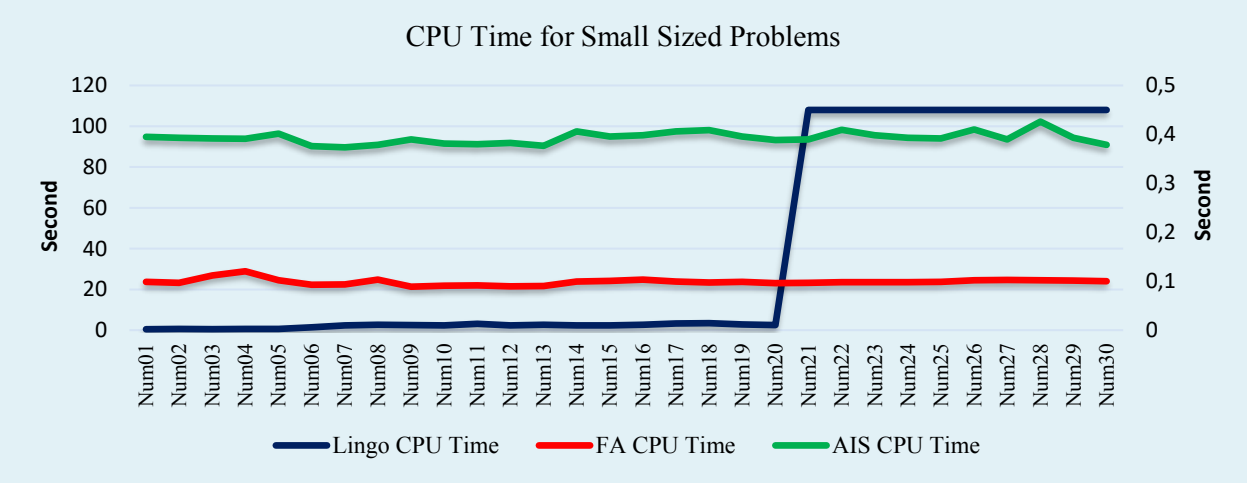

Fig.7. CPU times reported by Lingo 9.0 for AIS and FA in small problems

To create a more challenging comparison among proposed algorithms, 20medium-size and 20 large-size test problems are generated and solved seven times by each algorithm.

Table 6 shows computational results obtained by proposed algorithms, including AIS and FA, for medium-size and large-size problems.

Fig. 8 and Fig. 9 show average solutions obtained from proposed AIS and FA in medium and large size problems.

As can be seen from Fig. 8 and Fig 9., AIS yields better average solutions in both medium and large sizes problems

Table 6. Computational results for medium-size and large-size problems

Name Problem	AIS			FA		
	Average	Best	CPU times	Average	Best	CPU times
Num 31	179.38	179.38	19.05	196.33	181.52	62.24
Num32	183.67	183.67	19.18	209.64	200.2	61.62
Num33	193.29	193.29	19.32	216.93	200.12	60.63
Num34	165.42	165.42	19.61	182.58	178.54	60.97
Num35	173.36	173.36	19.66	189.04	179.93	60.29
Num 36	158.86	158.86	19.48	176.62	173.39	62.82
Num37	125.78	125.78	19.31	139.04	133.79	61.66
Num38	144.94	144.94	19.2	161.02	147.31	64.14
Num39	151.43	151.43	19.11	167	162.07	63.79
Num40	144.07	144.07	19.13	158.53	151.03	64.11
Num41	204.99	204.99	21.91	233.25	226.8	74.31
Num42	215.6	215.6	21.86	243.83	235.4	71.81
Num43	229.05	229.05	22.8	253.49	246.49	85.86
Num44	229.82	229.82	22.13	262.48	258.89	68.98
Num45	218.84	218.84	21.64	247.2	234.64	71.51
Num46	207.44	207.44	21.94	233.03	228.25	72.1
Num47	175.62	175.62	22.18	192.95	181.27	70.75
Num48	202.91	202.91	22.26	228.1	219.35	75.8
Num49	173.73	173.73	22.22	202.72	199.91	71.23
Num50	191.7	191.7	22.29	215.05	203.82	75.61
Num51	174.69	171.29	174.69	174.69	192.51	197.29
Num52	187.92	185.89	187.92	187.92	205.66	211.67
Num53	223.85	219.35	223.85	223.85	242.08	247.23
Num54	178.85	175.85	178.85	178.85	196.43	200.53
Num55	191.19	184.84	191.19	191.19	210.79	218.72
Num56	164.07	160.44	164.07	164.07	179.62	184.71
Num57	183.97	179.64	183.97	183.97	204.83	207.79
Num58	166.22	163.01	166.22	166.22	179.54	187.26
Num59	174.04	170.91	174.04	174.04	193.83	196.75
Num60	186.18	184.95	186.18	186.18	194.7	206.97
Num61	273.49	269.13	273.49	273.49	300.82	314.03
Num62	274.25	269.75	274.25	274.25	307.24	316.04
Num63	272.43	266.36	272.43	272.43	306.78	311.08
Num64	271.11	263.6	271.11	271.11	270.87	287.23
Num65	283.62	280.13	283.62	283.62	321.52	329.77
Num66	256.61	252.75	256.61	256.61	292.91	296.87
Num67	234.89	230.14	234.89	234.89	265.8	272.19
Num68	241.67	235.85	241.67	241.67	272.9	276.95
um69	244.5	238.79	244.5	244.5	278.08	284.67
Num70	238.89	236.26	238.89	238.89	269.16	279.4

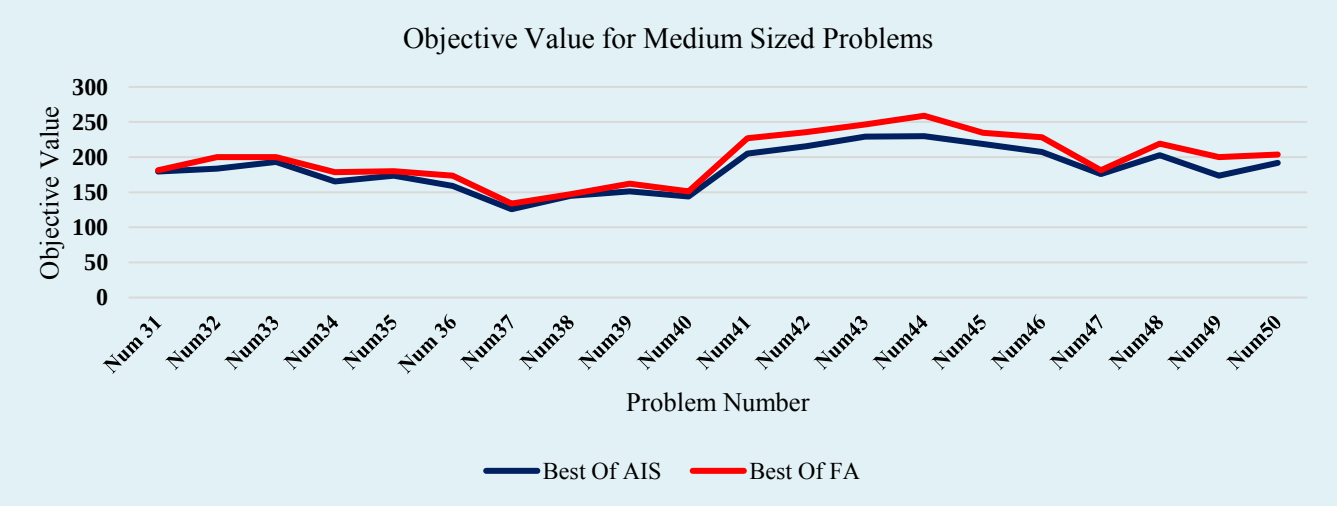

Fig.8. Comparing the average solution between AIS and FA for medium

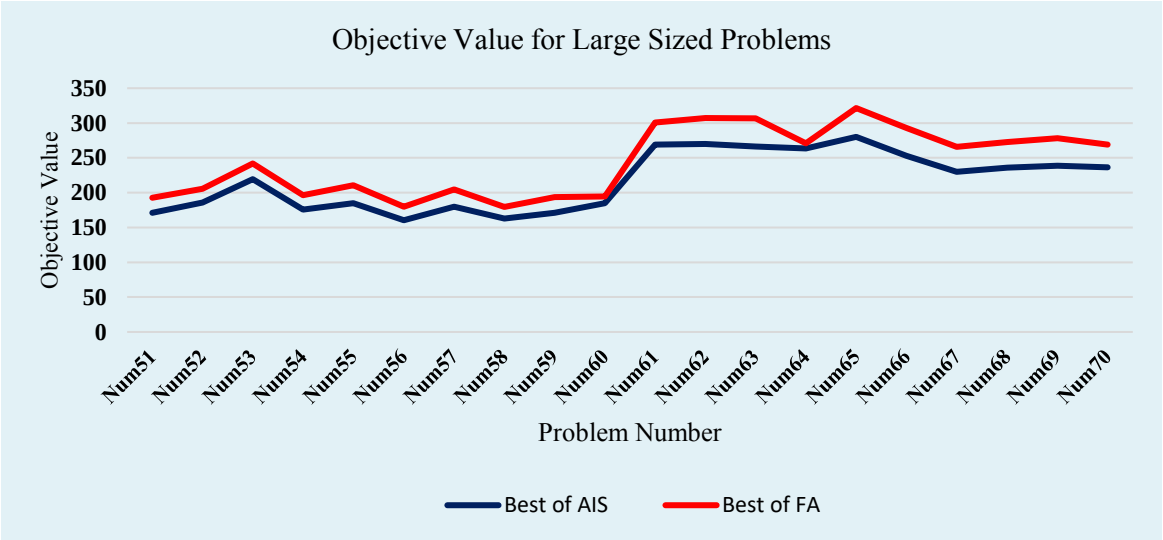

Fig.9. Comparing the average solution between AIS and FA for medium

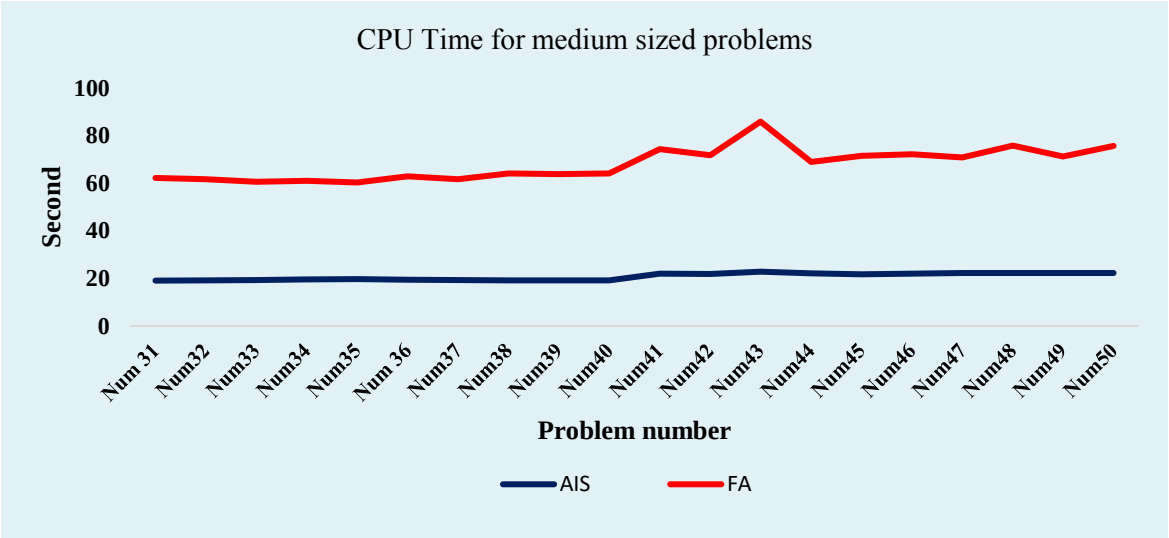

Fig 10. Comparing the average run time between AIS and FA for medium

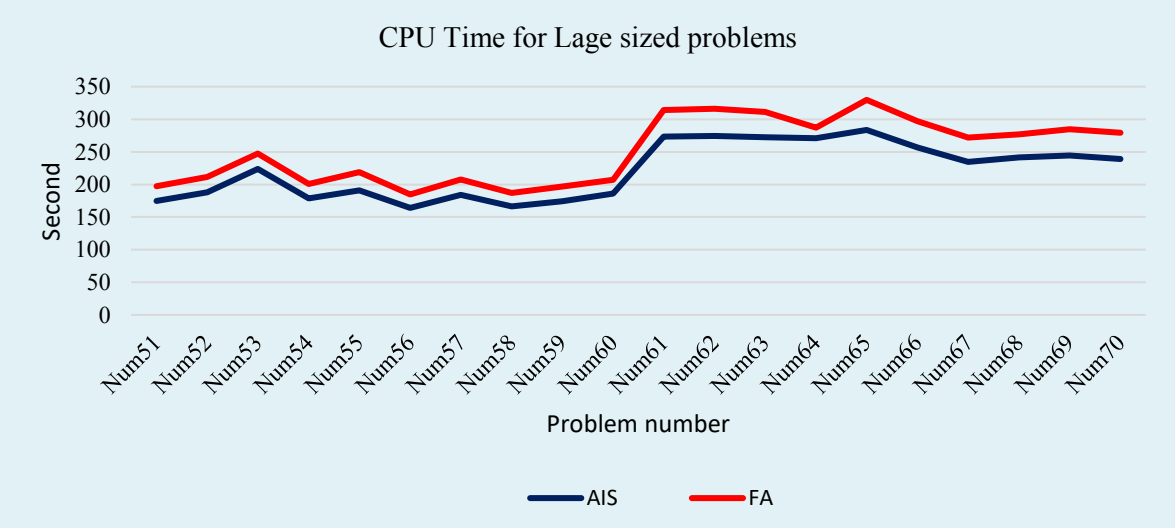

Fig 11. Comparing the average run time between AIS and FA for large problems

Fig. 10 displays the average CPU times obtained from the proposed AIS and FA for medium and large sizes problems.

As it can be noticed in Fig. 9 and Fig 10, AIS algorithm spends less CPU time than FA for achieving a near-optimal solution for medium and large size problems. Besides, the performance of algorithms is compared with each other based on the relative percentage of deviation (RPD).

RPD for each instance is calculated as follows:

$$\mathrm{RPD} = \frac{C^A - C^*}{C^*} * 100$$

Where C^A is the objective value for each run and $C^*$is the best value of each instance in seven run times. In small scales, $c^*$is the optimal solution obtained from Lingo 9.0 software. For medium and large scales, $c^*$is the best value obtained from both algorithms. The average RPD (ARPD) for all test problems is demonstrated in Table 7.

Table 7. ARPD of AIS and FA for all test problems

Problem Name	ARPD		Problem Name	ARPD		Problem Name	FA	
	AIS	FA		AIS	FA		AIS	FA
Num 01	1%	1%	Num 26	1%	4%	Num 51	2%	15%
Num 02	2%	4%	Num 27	2%	11%	Num 52	1%	14%
Num 03	4%	4%	Num 28	9%	21%	Num 53	2%	13%
Num 04	2%	4%	Num 29	2%	11%	Num 54	2%	14%
Num 05	2%	7%	Num 30	6%	18%	Num 55	3%	18%
Num 06	8%	14%	Num31	3%	9%	Num 56	2%	15%
Num 07	5%	5%	Num 32	2%	14%	Num 57	2%	16%
Num 08	0%	5%	Num 33	3%	12%	Num 58	2%	15%
Num 09	1%	7%	Num 34	3%	10%	Num 59	2%	15%
Num10	3%	4%	Num 35	3%	9%	Num 60	1%	12%
Num11	2%	5%	Num 36	3%	11%	Num 61	2%	17%
Num12	9%	17%	Num 37	2%	11%	Num 62	2%	17%
Num13	5%	6%	Num 38	2%	11%	Num 63	2%	17%
Num14	2%	8%	Num 39	3%	10%	Num 64	3%	17%
Num15	2%	4%	Num 40	4%	10%	Num 65	1%	18%
Num 16	1%	7%	Num 41	3%	14%	Num 66	2%	17%
Num17	0%	4%	Num 42	3%	13%	Num 67	2%	18%
Num18	7%	11%	Num 43	1%	11%	Num 68	2%	17%
Num19	2%	17%	Num 44	4%	14%	Num 69	2%	19%
Num20	3%	4%	Num 45	3%	13%	Num 70	1%	18%
Num21	2%	9%	Num 46	2%	12%			
Num22	4%	8%	Num 47	2%	10%			
Num23	4%	8%	Num 48	3%	12%			
Num24	1%	9%	Num 49	3%	17%			
Num25	2%	9%	Num 50	3%	12%			

To signify the difference between algorithms about statistical analysis, 95% confidence intervals for ARPD values are calculated for all test problems. Fig. 12 indicates the means plot and least significant difference intervals at a 95% confidence level for the ARPD of each algorithm in all test problems.

As can be seen in Fig.12, there is a statistically significant difference between AIS and FA algorithms. As there is no overlapping between confidence intervals of algorithms and AIS is closer to zero, AIS is more reliable than FA in solving the considered problem.

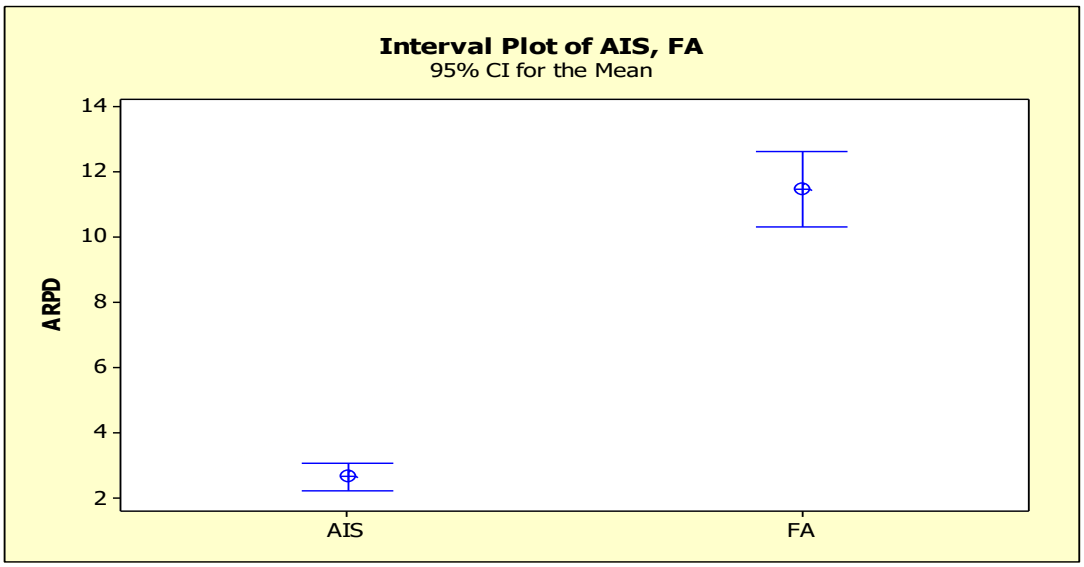

Fig 12. Means plot and LSD intervals (at 95% confidence level) for algorithms.

4.2 Practical Example

One striking application of the proposed problem, which is considered in this study, appears in the optimization of the distribution system of a Kalleh company. Kalleh is one of the food manufacture company which is located in Mazandaran province, north of Iran. Kalleh produces numerous products such as meat products (e.g., hamburgers, sausages, and frozen food), dairy products (e.g., pasteurized products, ultra-high temperature products, cheeses, ice cream, and sauces) and beverage products (e.g., non-alcoholic beer).

These products require a specific distribution system that needs some elements for distribution, such as salesmen, machines, and tools. This company's distribution sector has classified customers divided into three groups, including B2B customers, B2C, and B2W.

In this research, we consider a part of Kalleh's distribution system, which includes B2B customers in Amol city. In this case, the products should be delivered to 160 customers by four salesmen. In order to develop the distribution system and balance the workload between salesmen, initially, per customer has a particular grade based on its purchase, which is shown in Table 8. In the second step, customers should be assigned to salesmen by considering their grades and the distance between customers.

This is done in a way that the total revenue of each salesman
as a portion of total sales is maximized. The problem is formulated according to our proposed model and solved by recommended algorithms. The results are shown in Table 9. It can be concluded from Table 9 that the proposed AIS has found a better solution than FA, as the completion times of tours are shorter and have lower deviations.

Table 8. Grade of customers

Grade of customer	Number of customers
A	1-2-3-5-8-14-15-20-25-33-37-40-41-51-54-70-75-87-88-94-107-116-118-135-137-138-144
B	4-9-10-12-13-17-21-27-28-30-31-34-38-39-42-46-49-53-55-56-64-65-67-69-71-72-76-82-84-85-86-90-93-95-96-98-104-106-111-114-115-119-125-128-130-131-132-136-139-140-141-143-151-154-155-156-157-158-159-160
C	6-7-11-16-18-19-22-23-24-26-29-32-35-36-43-44-45-47-48-50-52-57-58-59-60-61-62-63-66-68-73-74-77-78-79-80-81-83-89-91-92-97-99-100-101-102-103-105-108-109-110-112-113-117-120-121-122-123-124

Table 9. Results of solving the Practical example

AIS			FA		
Salesmen numbers	number of customers assigned to each salesman	completion time of each tour	Salesmen numbers	number of customers assigned to each salesman	completion time of each tour
1	44	839.217	**1**	44	1095/895
2	47	847.112	**2**	38	1088/278
3	41	838.905	**3**	45	1099/256
4	29	835.513	**4**	34	1081/326

According to the obtained results from AIS algorithm, the sequence of assigned fast food stores to each salesman are as follows:

- **Salesman 1** visits fast foods (150,118,117,90,85,20,15,131,5,107,18,130, 127,79,72,6,143,135,9,27,133,116,125,124, 106,59,52,58,51,57,69,68,120,14,149,109,2, 110,13,25,139,140,136,35,11).
- **The salesman 2** visits fast foods (129,145,102,34,74,67,60,71,77,83,99,37,3, 61,66,65,21,24,148,157,147,146,95,101,41, 86,78,29,98,159,156,141,94,100,28,23,62 63).
- **The salesman 3** visits fast foods (84,82,92,32,122,121,126,114,49,48,43,103, 88,96,119,73,53,54,70,64,47,56,55,1,87,8,1 51,4,10,16,26,128,115,111,7,75,44).
- **The salesman 4** visits fast foods (108,50,46,105,113,91,89,45,39,38,42,40,97 ,104,112,30,153,160,144,155,161,142,31,76 ,80,81,36,93,137,138,134,152,17,12,123,15 8,154,33,132,19,22).

5 Discussions

From the experimental results, both algorithms obtained the solution of exact methods in most cases of small instances. The AIS algorithm can provide much shorter executing time even in these small sized problems. Based on the Table 6 in all medium and large sized instances, AIS can yield shortest maximum completion time for which can bring better work load balancing among all visitors in less run time in comparison with FA algorithm. Moreover, by statistical analysis of the solution of both algorithm less variation in objective value of AIS is presented and this algorithm is able to escape from local optimal solution.

6 Conclusion

This research presents a model of multi-travel salesman problem which is done by considering customers with distinct grades, different salesmen with varied skills which are influenced by the effect of learning on salesman's skill, in order to balance salesmen workload and minimizing the makespan. According to the complexity degree of a problem, this study has suggested two meta-heuristic algorithms, including AIS and FA. In order to achieve a robust design of the proposed algorithm, parameters were tuned based on the Taguchi method. Furthermore, some computational experiments were carried out to evaluate the efficiency of the proposed algorithms. This is applied through three different sets of problems in scales of small, medium, and large. Comparisons have revealed that both proposed meta-heuristics are effective and efficient in solving small-scale problems. However, in medium and large scales problems, AIS had a better performance against FA. Also, AIS produced a more optimal or near-optimal solution than FA with a lesser average time. Subsequently, the applicability and capability of the proposed framework are tested on a case study in a specific company named Kalleh. As future works, it is fascinating to improve the meta-heuristics by hybridizing them with other efficient heuristic or meta-heuristic algorithms. It is also impressive to extend the problem by adding some reasonable assumptions, including time windows and multiple depots.

References:

[1] Bektaş, T. (2012). Formulations and Benders decomposition algorithms for multi depot salesmen problems with load balancing. European Journal of Operational Research, 216(1), 83-93.

[2] Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega, 34(3), 209-219.

[3] Svestka, J. A., & Huckfeldt, V. E. (1973). Computational experience with an m-salesman traveling salesman algorithm. *Management Science*, *19*(7), 790-799.

[4] Bellmore, M., & Hong, S. (1974). Transformation of multisalesman problem to the standard traveling salesman problem. *Journal of the ACM (JACM)*, *21*(3), 500-504.

[5] Russell, R. A. (1977). An effective heuristic for the m-tour traveling salesman problem with some side conditions. *Operations Research*, *25*(3), 517-524.

[6] Hong, S., & Padberg, M. W. (1977). A note on the symmetric multiple traveling salesman problem with fixed charges. *Operations Research*, *25*(5), 871-874.

[7] Tang, L., Liu, J., Rong, A., & Yang, Z. (2000). A multiple traveling salesman problem models for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *European Journal of Operational Research*, *124*(2), 267-282.

[8] Qu, H., Yi, Z., & Tang, H. (2007). A columnar competitive model for solving multi-traveling salesman problem. *Chaos, Solitons & Fractals, 31(4), 1009-1019.*

[9] Yadlapalli, S., Malik, W. A., Darbha, S., & Pachter, M. (2009). A Lagrangian-based algorithm for multiple depots, multiple traveling salesmen problems. *Nonlinear Analysis: Real World Applications*, *10*(4), 1990-1999.

[10] Ghafurian, S, Javadian, N, an ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems, *Applied Soft Computing 11(1) (2011) 1256-1262.*

[11] Shim, V. A., Tan, K. C., & Cheong, C. Y. (2012). A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(5), 682-691*.

[12] Yuan, S., Skinner, B., Huang, S., & Liu, D. (2013). A new crossover approach for solving the multiple traveling salesmen problem using genetic algorithms. *European Journal of Operational Research*, *228*(1), 72-82.

[13] Mohammad pour, T., & Yadollahi, M. (2014). Solving the problem of multiple traveling salesman problems using a hybrid gravitational algorithm. *International Journal of Communications, 3, 32-37.*

[14] Soylu, B. (2015). A general variable neighbourhood search heuristic for multiple traveling salesmen problem. *Computers & Industrial Engineering*, *90*, 390-401.

[15] Kota, L., & Jarmai, K. (2015). Mathematical modelling of multiple tour multiple traveling salesman problems using evolutionary programming. *Applied Mathematical Modelling*, *39*(12), 3410-3433.

[16] Changdar, C., Pal, R. K., & Mahapatra, G. S. (2017). A genetic ant colony optimization-based algorithm for solid multiple traveling salesmen problem in a fuzzy rough environment. Soft Computing, 21(16), 4661-4675.

[17] Jozefowiez, N., Semet, F., & Talbi, E. G. (2008). Multi-objective vehicle routing problems. *European journal of operational research*, *189*(2), 293-309.

[18] Jozefowiez, N., Semet, F., & Talbi, E. G. (2009). An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, *195*(3), 761-769.

[19] Lee, T. R., & Ueng, J. H. (1999). A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution & Logistics Management*, *29*(10), 646-657.

[20] Ribeiro, R., & Ramalhinho Dias Lourenço, H. (2001). A multi-objective model for a multi-period distribution management problem

[21] Lacombe, P., Prins, C., & Ramdane-Cherif, W. (2004). Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, *131*(1-4), 159-185.

[22] Jozefowiez, N., Semet, F., & Talbi, E. G. (2005, October). Enhancements of NSGA II and its application to the vehicle routing problem with route balancing. In *International Conference on Artificial Evolution (Evolution Artificielle)* (pp. 131-142). Springer, Berlin, Heidelberg.

[23] Lacomme, P., Prins, C., Prodhon, C., & Ren, L. (2015). A multi-start split based path relinking (MSSPR) approach for the vehicle routing problem

with route balancing. *Engineering Applications of Artificial Intelligence*, *38*, 237-251.

[24] Venkatesh, P., & Singh, A. (2015). Two metaheuristic approaches for the multiple traveling salesperson problems. *Applied Soft Computing*, *26*, 74-89.

[25] Ajam .M, Akbari. V, Salman.F. Minimizing latency in post-disaster road clearance operations. *Eur. J. Oper. Res. 277(3): 1098-1112 (2019)*

[26] Mir, M. S. S., & Rezaeian, J. (2016). A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines. *Applied Soft Computing*, *41*, 488-504.

[27] Wu, Y. B., Wang, X. Y., & Ji, P. (2012). A note on "Single-machine scheduling problems with both deteriorating jobs and learning effect." *Applied Mathematical Modelling*, *36*(12), 6341-6344.

[28] Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research*, *115*(1), 173-178.

[29] Cheng, T. E., & Wang, G. (2000). Single machine scheduling with learning effect considerations. *Annals of Operations Research*, *98*(1-4), 273-290.

[30] Li, X., Wang, J., Zhou, J., & Yin, M. (2010, July). An effective GSA based memetic algorithm for permutation flow shop scheduling. In *Evolutionary Computation (Cec), 2010 Ieee Congress On* (pp. 1-6). IEEE.

[31] Lee, W. C. (2011). Scheduling with general position-based learning curves. *Information Sciences*, *181*(24), 5515-5522.

[32] Wu, Y. B. & Wang, J. J. (2016). Single-machine scheduling with truncated sum-of-processing-times-based learning effect, including proportional delivery times. *Neural Computing and Applications*, *27*(4), 937-943.

[33] Mir, M. S. S., & Rezaeian, J. (2016). A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines. *Applied Soft Computing*, *41*, 488-504.

[34] Ahmadvand, M., Yousefikhoshbakht, M., & MAHMOODI, D. N. (2012). Solving the traveling salesman problem by an efficient hybrid metaheuristic algorithm.

[35] Brown, E. C., Ragsdale, C. T., & Carter, A. E. (2007). A grouping genetic algorithm for the multiple traveling salesperson problems. *International Journal of Information Technology & Decision Making*, *6*(02), 333-347.

[36] Lin, S. W., & Ying, K. C. (2013). Minimizing makespan in a blocking flow shop using a revised artificial immune system algorithm. Omega, 41(2), 383-389.

[37] Zandieh, M., Ghomi, S. F., & Husseini, S. M. (2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Applied Mathematics and Computation, 180(1), 111-127.

[38] Afzalirad, M., & Rezaeian, J. (2016). Resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times, precedence constraints, and machine eligibility restrictions. Computers & Industrial Engineering, 98, 40-52.

[39] Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flow shop scheduling problem. European Journal of Operational Research, 177(3), 2033-2049.

[40] Fister, I., Fister Jr, I., Yang, X. S., & Brest, J. (2013). A comprehensive review of firefly algorithms. Swarm and Evolutionary Computation, 13, 34-46.

[41] X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver, Press UK (2008)

[42] Yang, X. S. (2009, October). Firefly algorithms for multimodal optimization. *In International symposium on stochastic algorithms (pp. 169-178). Springer, Berlin, Heidelberg.*

[43] Łukasik, S., & Żak, S. (2009, October). Firefly algorithm for continuous constrained optimization tasks. In *International conference on collective computational intelligence* (pp. 97-106). Springer, Berlin, Heidelberg.

[44] Chen, H., Li, S., & Tang, Z. (2011). Hybrid gravitational search algorithm with a random-key encoding scheme combined with simulated annealing. IJCSNS, 11(6), 208.

[45] Li, X., Wang, J., Zhou, J., & Yin, M. (2010, July). An effective GSA based memetic algorithm for permutation flow shop scheduling. In *Evolutionary Computation (Cec), 2010 Ieee Congress On* (pp. 1-6). IEEE.

[46] Taguchi, G. (1986). Introduction to quality engineering: designing quality into products and processes (No. 658.562 T3).

[47] Afzalirad, M., & Rezaeian, J. (2017). A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approach. *Applied Soft Computing*, *50*, 109-123.

Contribution of individual authors to the creation of a scientific article

All steps of this research have been done under supervision of **Chiara.Colombaroni** ranging from problem definition and finding gap of research to optimization and analysing of problem.

Mostafa Mohammadi has formulated mathematical model and implemented the Algorithm FA, parameter tuning of algorithms.

Golman Rahmanifar has implemented the AIS Algorithm and applied exact approach of optimization in Lingo and organized and executed the experimental problems.