Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 13, Number 9 (2017), pp. 4747-4764
© Research India Publications
http://www.ripublication.com

Solving multi objective Assignment problem using Tabu search algorithm

Abdul Munaam Kadhem Hammadi

Department of Statistics, College of Economic & Administration, University of Baghdad, Baghdad, Iraq.

Abstract

This paper presents the multi-objective tabu search method for the multi-objective assignment problem. As a well-known adaptation of the tabu search, it is used heuristically to generate non-dominated alternatives to multi-objective combinatorial optimization problems. MOTS works with a set of current solutions that, thanks to the manipulation of weights, are optimized towards the non-dominated border while trying to disperse on the border. It generated some problem to measure the effectiveness of the algorithm three different objective sizes of the question and compare the results with the simulated annealing algorithm and the genetic algorithm and particle swarm optimization by three measurements. The algorithm has shown great efficiency in solving the problem in relation to the rest of the roads.

Keywords Combinatorial optimization , Multi-objective optimization, Tabu search, Assignment problem ,Particle Swarm optimization .

1. INTRODUCTION

The classical assignment problem (AP) is to find a one-to one matching between n jobs and n workers, the objective being to minimize the total cost or to maximize the total efficiency of the assignments. The classical AP has been extensively studied in the literature and many algorithms are available to solve it, such as the Hungarian method [5], the flow type method [3], the Munkres method [9], etc.

In the real world problems are most of The assignment problem contain multiple objective such as cost reduction while at the same time and in fact more difficult problem increasing the number of objectives and turn from the problem of multi-algorithm solution of the border into the problem more complex is geometry as most issues multiple objective does not have an polynomial algorithm And be in the field of multi-objective combinatorial optimization problem.

2. THE MULTI OBJECTIVE OPTIMIZATION PROBLEM

The problem of multi-objective optimization can then be defined as the problem of finding a vector of decision variables that satisfies the constraints and optimizes a vector function whose elements represent the objective functions. These functions are a mathematical description of the performance criteria that are usually in conflict with each other. Therefore, the term "optimize" means finding such a solution that would give the values of all objective functions acceptable to the decision maker.

The general problem of multi-objective optimization can now be formally defined as follows:

Definition 1 (General MOP) : Find the vector $\vec{x}^* = [x_1^*, x_2^*, \dots, x_n^*]^T$ Which satisfies the m inequality constraints:

$$g_i(\vec{x}) \geq 0 \quad i = 1,2,\dots,m \qquad (1)$$

the p equality constraints

$$h_i(\vec{x}) \geq 0 \quad i = 1,2,\dots,p \qquad (2)$$

and optimizes the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})]^T$$

In other words, it aims to determine from the set of all the numbers that satisfy (1) and (2) the particular set $x_1^*, x_2^*, \dots, x_n^*$ which gives the values Optimal all-purpose functions. The constraints given by (1) and (2) define the feasible region n and any

The point $\vec{x}$ in Ω defines a workable solution. The vector function $\vec{f}(\vec{x})$ is a function that maps the set Ω into the set Λ which represents all the possible values of the objective functions. The k components of the vector $\vec{f}(\vec{x})$represent the noncommensurable criteria that must be considered. The constraints $g_i(\vec{x})$ and $h_i(\vec{x})$represent the restrictions imposed on the decision variables. The vector i* is reserved to denote the optimal solutions (there are usually more than one).

3. MULTI OBJECTIVE ASSIGNMENT PROBLEM

The general multi-objective combinatorial optimization problem can be expressed as:

$$(MOCO)\begin{cases} \min F(x) = \left(f^1(x), f^2(x), \dots f^p(x)\right) \\ \qquad x \in S \end{cases}$$

Where $p \geq 2$ is the number of objective functions, $x = (x_1, x_2, \ldots, x_d)$ is the vector representing the decision variables, S is the (finite) set of feasible solutions in the solution space $\mathbb{R}^d$. The set Z = F(S) represents the feasible points (outcome set) in the objective space $\mathbb{R}^p$ and $z = (z^1, z^2, \ldots, z^p)$, with $z^i = f^i(x)$, is a point of the objective space.

Note that, in (MOCO), the term "min" appears in quotation marks because, in general, there does not exist a single solution that is minimal on all objectives. As a consequence, several concepts must be established to define what an optimal solution is. The more used one is the dominance relation (also known as Pareto dominance) (Fig. 1):

Definition 2 A point $z = (z_1, z_2, \ldots, z_p)$ dominates a point $w = (w_1, w_2, \ldots, w_p)$ and we write $z \leq w$ if and only if for all i ∈ $\{1, \ldots, p\}$, $zi \leq wi$ with for at least one $j \in \{1, \ldots, p\}$ such that $(zj < wj)$.

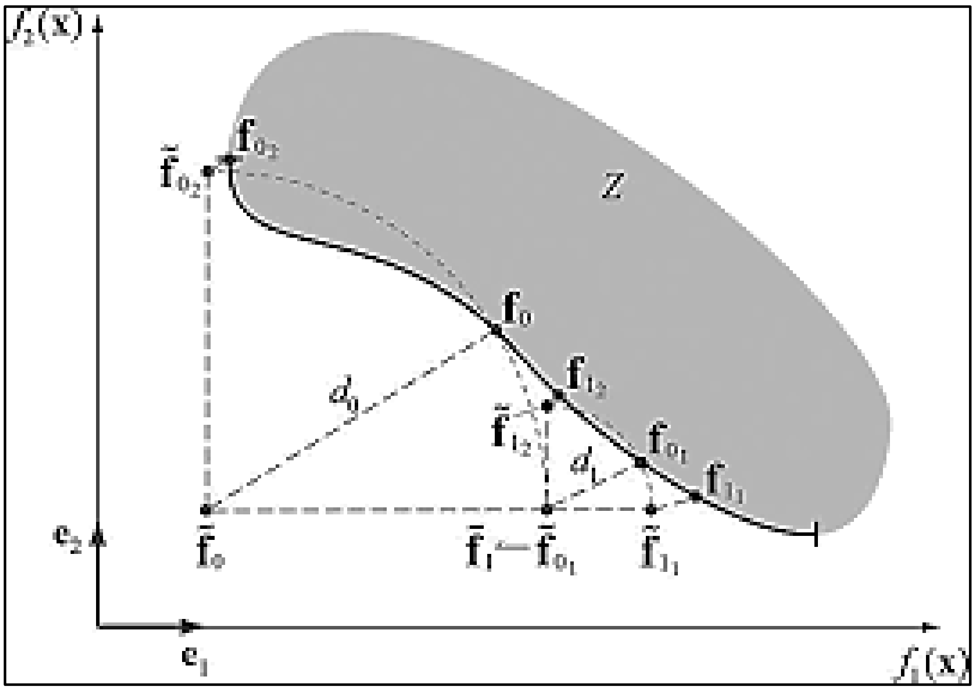

Fig. 1 Dominations in the Pareto sense in a bi-objective Space

Definition 3 A solution x∗ ∈ S is called (Pareto) efficient for (MOCO) if and only if there does not exist any other feasible solution $x \in S$, such that $f_i(x) \leq f_i(x^*)$ for $i = 1..p$ with at least one $j \in \{1, \ldots, p\}$ such that $f_j(x) \leq f_j(x^*)$. The point $F(x^*)$

is then called a non-dominated point. The set of efficient solutions, also called the Pareto optimal set, is often denoted by E and the image of E in Z is called the non-dominated frontier or the Pareto optimal front, and is denoted by Z_E.

The single objective assignment problem (AP) is an integer programming problem that can be solved as a linear program due to total unimodularity of the constraint matrix. Efficient algorithms to solve it, e.g., the Hungarian method or the successive shortest paths method (Papadimitriou and Steiglitz, 1982; Ahuja et al., 1993) are well known. In this paper, we consider the assignment problem with two objectives (BAP). It can be formulated as follows:

$$F(\varphi) = \sum_{k=1}^{p} \sum_{i=1}^{n} C^{k}{}_{i,\varphi(i)} \qquad (3)$$

The solution can be found through many ways, where the solution is presented by a permutation vector φ, of n elements:

$$\varphi = \{\varphi_1, \varphi_2, \dots, \varphi_n\}$$

The difficulty of MOCO problems results from the following two factors.

• The resolution of an MOCO problem requires intensive cooperation with the decision maker (DM), which translates into particularly high requirements for the effective tools used to generate effective solutions.

• Many combinatorial problems are difficult, even in single-objective versions, their multiple objective versions are often more difficult.

In addition, many single-purpose combinatorial problems belong to the class of NP-hard problems. Generating effective solutions in an MOCO problem is obviously no easier than finding solutions that optimize particular objectives and in many cases is even more difficult. For example,

The problem of the shortest path of a single objective is one of the simplest combinatorial problems, whereas the problem corresponding to several objectives is NP-hard (Garey and Johnson, 1979). Thus, even problems where accurate methods of a single relatively efficient objective are known may be difficult if several objectives are to be considered.

The tools used to generate effective solutions in MOCO, such as single-purpose optimization methods, can be categorized as one of the following:

• exact procedures;

• specialized heuristic procedures;

• meta-theorist procedures.

4. LOCAL SEARCH ALGORITHMS

In this section, we describe the metaheurists called trajectory methods. The term trajectory methods is used because the search process performed by these methods is characterized by a trajectory in the search space. Consequently, a successive solution may or may not be in the vicinity of the present solution. The process of searching for trajectory methods can be considered as the (discrete) evolution of a discrete dynamic system (Bar-Yam 1997; Devaney 1989]. The algorithm starts from an initial state (the initial solution) and describes a trajectory in the state space. The dynamics of the system depend on the strategy used; Simple algorithms generate a trajectory composed of two parts: a transitory phase followed by an attractor (a fixed point, a cycle or a complex attractor). Algorithms with advanced strategies generate more complex

```
Procedure basic LS
$s \leftarrow GenerateInitialSolution()$
  $\boldsymbol{repeat}$
    $s \leftarrow Improve(N(s))$
$\boldsymbol{until}$ $no\ improvement\ is\ possible$
```

Fig. 2. Algorithm: Iterative Improvement.

Trajectories that can not be subdivided into these two phases. The characteristics of the trajectory provide information about the behavior of the algorithm and its effectiveness with respect to the instance addressed. It should be emphasized that the dynamics is the result of the combination of the algorithm, the representation of the problems and the instance of the problem. In fact, the representation of the problem and the structures of the neighborhood define the research landscape; The algorithm describes the strategy used to explore the landscape and finally the actual characteristics of the search space are defined by the instance of the problem to be solved. We will first describe the basic local search algorithms before moving on to more complex strategies. Finally, we deal with algorithms that are general exploration strategies that can integrate other trajectory methods as components.

4.1 Basic Local Search: Iterative Improvement

Basic local search is usually referred to as iterative improvement, because each movement is only performed if the resulting solution is better than the current solution. The algorithm stops as soon as it finds a local minimum. The high level algorithm is outlined in Figure 2. The Enhance (N (s)) function can be either at the extreme or at a first improvement, or at any intermediate option. The first analyzes the N (s) of the neighborhood and chooses the first

A better solution than s, the latter exhaustively explores the neighborhood and returns one of the solutions with the lowest objective function value. Both methods stop at local minimums. Therefore, their performance strongly depends on the definition of $S, f\ and\ N$. The performance of Iterative improvement procedures on COP is generally quite unsatisfactory. Consequently, several techniques have been developed to prevent algorithms from being trapped in local minima, which is done by adding mechanisms that allow them to escape local minima. This also implies that the termination conditions of the metaheurstic algorithms are more complex than simply reaching a local minimum.. Indeed, the possible termination conditions include: the maximum CPU time, a maximum number of iterations, a solution s with $f\ (s)$ less than a predefined threshold value or the maximum number of iterations without improvements is reached.

4.2 Simulated Annealing

Simulated annealing (SA) is generally considered to be the oldest among the metaheurstic and certainly one of the first algorithms that had an explicit strategy to escape the local minima. The origins of the algorithm are in statistical mechanics (Metropolis algorithm) and it was first presented as a search algorithm for COP in Kirkpatrick et al. [1983] and Cerny [1985]. The fundamental idea is to allow movements resulting in quality solutions worse than the current solution (uphill movements) in order to escape the local minima. The probability of such a movement is diminished during the search. The high level algorithm is described in Figure 3.

```
Procedure basic SA
$s \leftarrow GenerateInitialSolution()$
$T \leftarrow T0$
$\textbf{while}$ $termination\ conditions\ not\ met$ $\textbf{do}$
$s' \leftarrow PickAtRandom(N(s))$
$\textbf{if}$ $(f\ (s') < f\ (s))$ $\textbf{then}$
$s \leftarrow s'$          $\%\ s'\ replaces\ s$
$\textbf{else}$
$Accept\ s'\ as\ new\ solution\ with\ probability\ p(T, s', s)$
$\textbf{endif}$
$Update(T)$
$\textbf{endwhile}$
```

Fig. 3. Algorithm: Simulated Annealing (SA).

The algorithm begins by generating an initial solution (constructed in a random or heuristic manner) and the initialization of the temperature parameter T. Then, at each iteration, a solution $s' \in N(s)$ is randomly sampled and is accepted as F (s), f (s), and T. s0 replaces s if $f(s') < f(s)$ or, in the case $f(s')$ ¸ f Which is a function of T and $f(s') - f(s)$. The probability is generally computed following the Boltzmann distribution $e^{(-\frac{f(s\prime)-f(s)}{T})}$.

The temperature T decreases during the search process. Thus, at the beginning of the search, the probability of accepting uphill movements is high and it gradually decreases, converging towards a simple iterative improvement algorithm. This process is analogous to the metal and glass annealing process, which provides a low energy configuration when cooled with a suitable cooling program. Regarding the search process, this means that the algorithm is the result of two combined strategies: random walk and iterative improvement. In the first phase of the research, the bias of the improvements is low and allows to explore the research space; This erratic component decreases slowly, which leads the search to converge to a minimum (local). The probability of accepting uphill movements is controlled by two factors: the difference between objective functions and temperature. On the one hand, at a fixed temperature, the higher the difference $f(s') - f(s)$, the lower the probability of accepting a displacement from S to S'. On the other hand, the higher the T, the higher the probability of uphill climb. Choosing an appropriate cooling schedule is crucial to the performance of the algorithm. The cooling program defines the value of T at each iteration k,

$T(k+1) = Q(T_k, k)$, where $Q(T_k, k) = \alpha T_k$ is a function of temperature and number of iterations and $\alpha \in (0,1)$ To an exponential decay of the temperature ... Theoretical results on non-homogeneous Markov chains [Aarts et al. 1997] indicate that, under particular conditions on the cooling schedule, the algorithm converges in probability to a global one.

Water cooling can be used to reduce the intensity of the water. For example, at the beginning of the search, T could be constant or decrease linearly, in order to sample the search space; Then T could follow a rule such as geometry, to converge to the minimum at the end of the search.

More successful variants are non-monotonic cooling schedules (see, for example, Osman [1993] and Lundy and Mees [1986]). Non-monotonic cooling schedules are characterized by alternating phases of cooling and reheating, which provide an oscillating balance between diversification and intensification.

The cooling system and the initial temperature should be adapted to the particular problem because the cost of exhausting local minima depends on the structure of the research landscape. A simple way of empirically determining the starting temperature T_0 is to first sample the search space with a random walk to approximate the mean and the variance of the values of the objective function. But also more elaborate schemes can be implemented [Ingber 1996].

4.3 Tabu search

Tabu Search (TS) is one of the most cited metaheuristics and used for CO problems. The basic ideas of TS were first introduced in Glover [1986], based on earlier ideas in Glover [1977].

The method and its concepts can be found in Glover and Laguna [1997]. TS explicitly uses the search history, both to escape local minima and to implement an exploration strategy. We will first describe

A simple version of TS, to present the basic concepts. Then we will explain a more applicable algorithm and finally we will discuss some improvements.

```
Procedure basic TS
$s \leftarrow GenerateInitialSolution()$
$TabuList \leftarrow \phi$
$while\ termination\ conditions\ not\ met\ do$
$s \leftarrow ChooseBestOf(N(s) \setminus TabuList)$
$Update(TabuList)$
$endwhile$
```

Fig. 4. Algorithm: Simple Tabu Search (TS).

The simple TS algorithm applies a better local search for improvement as a basic ingredient and uses a short-term memory to escape local minima and to avoid cycles. The short-term memory is implemented in the form of a list of tables which allows to follow the last solutions visited and prohibits the displacements towards them. The neighborhood of the current solution is

So limited to solutions that do not belong to the list of tabuses. In the following, we will refer to this set as defined.

At each iteration, the best solution of the authorized game is chosen as the new current solution. Moreover, this solution is added to the list of tabu and one of the solutions already contained in the tabu list is deleted (usually in a FIFO command). Due to this dynamic restriction of permitted solutions in a neighborhood, TS can be considered as a dynamic neighborhood research technique [Stutzle 1999b]. The algorithm stops when a termination condition is met. It can also terminate if the allowed set is empty, if all solutions of N (s) are forbidden by the list of tabu. Using a tabu list prevents going back to recently visited solutions, Therefore, it prevents endless cycling and forces research to accept uphill movements. The length l of the tabu list (the duration of the tabu) controls the memory of the search process. With small tabu positions, the search will focus on small areas of the search space. On the contrary, a large regime of taboos forces the research process to explore larger regions because it forbids revisiting a greater number of solutions. The duration of the tabu may vary during the search, which leads to more robust algorithms. An example can be found in Taillard [1991], where the tabu regime is periodically reset randomly from the interval [lmin, lmax]. A more advanced use of a dynamic taboo regime is presented in Battiti and Tecchiolli [1994] and Battiti and Protasi [1997], where

The pension rate is increased if there is evidence of repetition of solutions (So greater diversification is needed), 8Strategies to avoid stopping the search when the allowed play is empty include choosing the less Solution recently visited, even if it is a tabu. 9Cycles of higher period are possible, since the list of tabu has a finite length l which is smaller than the Cardinality of the search space. While it decreases if there are no improvements (so intensification should be strengthened). More advanced ways to create a dynamic taboo regime are described in Glover [1990]. However, implementing short-term memory as a list containing complete solutions is not practical because managing a list of solutions is very inefficient. Therefore, instead of the solutions themselves, the solution attributes are stored. Attributes are usually solution components, movements, or differences between two solutions. Since more than one attribute can be considered, a tabu list is entered for each attribute. The set of attributes and the corresponding tabu lists define the tabu conditions that are used to filter the neighborhood of a solution and generate the allowed set. Storing attributes instead of complete solutions is much more efficient, but it introduces a loss of information because forbidding an attribute means affecting the state of the table to probably more than one solution. Thus, it is possible that non-visited solutions of good quality are excluded from the allowed play. To overcome this problem,

Suction criteria are defined that allow a solution to be included in the allowed game, even if it is prohibited by the tabu conditions. Suction Criteria

Set the suction conditions used to build the allowed set. The most commonly used aspiration criterion selects better than the best solutions. The complete algorithm, as described above, is reported in Figure 4. Taboo lists are only one way to take advantage of the search history. They are generally identified with the use of short-term memory. The information gathered throughout the search procedure can also be very useful, especially for a strategic orientation of the algorithm. This type of long-term memory is generally added to TS by referring to four principles: recurrence, frequency, quality and influence. Memory records based on recurrence for each solution (or attribute) the most recent iteration in which it was involved. The memory based on the orthogonal frequency keeps track of how often each solution (attribute) was visited. This information identifies the regions (or Subsets) of the solution space where the search was confined or where there remained a large number of iterations. This type of information about the past is usually exploited to diversify research. The third principle (that is, quality) refers to the accumulation and retrieval of information from the search history to identify good solution components. This information can be integrated efficiently into the construction of the solution. Other metaheuristics (for example, Ant Colony Optimization) explicitly use this principle to know the right combinations of components in the solution. Finally, influence is a property regarding the choices made during the search and can be used to indicate which choices are the most critical. In general, the TS field is a rich source of ideas. Many of these ideas and strategies have been and are being adopted by other metaheuristics. TS has been applied to most CO problems; Examples for successful applications are Robust Tabu Search in QAP [Taillard 1991], Tabu reactivity to the MAXSAT problem [Battiti and Protasi 1997], and assignment problems [Dell'Amico et al. 1999]. TS approaches dominate the problematic area of Job Shop Scheduling (JSS) (see, for example, Nowicki and Smutnicki [1996]) and the vehicle routing area (VR) [Gendreau et al. 2001]. Other current applications can be found on [Tabu Search 2003 website].

5. MULTI OBJECTIVE TABU SEARCH ALGORITHMS

The multi-objective tabu search procedure, MOTS, works with a set of current solutions that are simultaneously optimized towards the non-dominated border. The points of the current solutions are sought to cover the entire boundary and, for each solution, several times, an optimization direction is made so that it tends to move away from the other points in the direction of the boundary Not dominated. The solutions in turn take the application of a motion according to a tabu search heuristic and each solution retains its own tabu list. In the following, we will annotate the Pascal contour of the MOTS database in FIG. 5.

In row 1, each current solution is set to a random startup solution and the list of tabs (TL) is flushed. In line 2, the current set of non-dominated points (ND) is emptied, an iteration counter is reset and the range equalization factors (p) are defined on a unit vector. We then start the loop which continually leaves each current solution passing a neighboring solution until a certain STOP criterion is respected. In lines 5-11, the weight vector (l) for the point is determined. This vector belongs to L and thus guarantees the optimization towards the non-dominated limit. We want to repair the weights so that the point moves away from the other points, ideally, that the points are distributed equidistantly on the border. Therefore, each element of the weight vector is defined as a function of the proximity of other points for this purpose. However, we only compare one point with the points of the current solution to which it is not dominated. The closer a point is, the more it should influence the weight vector. Proximity is measured by a distance function (d) as a function of certain measurements in the objective function space and using the weights of the range. The influence is given by a decreasing and positive proximity function (g) over the distance. In practice, the well-functioning proximity function $(d) = 1/d$, as well as the Manhattan distance (used on the scale objectives by the beach equalization factors, i.e. $d(z_i, z_j, p) = \sum \pi^k |z_i^k - z_j^k|$ Emphasis on result In rows 12 to 15, the standard tabu search procedure is used to replace a current solution with the possible neighborhood solution (generated by the neighborhood function N) that can be reached from Tabu is a type of attribute (A) on solutions of solutions and solutions to avoid movements to the solution. The best neighbor is determined by the scalar product between the weight vector and the vector objective function.

```
Procedure basic MOTS
1. for each solution $x_i$ in X do set $x_i$ to a random feasible solution and set TLi $= \{\}$
2. set ND $= \phi$ and set count $= 1$ and set $\pi^k = 1/n$ for all objectives k
3. repeat
4. for each solution $x_i$ in X do
5. set $\lambda = 0$
6. for each solution j in X where f(xj)is n − dominated by f(xi) and f(xi) $\neq$ f(xj) do
7. set w $= g(d(f(xi), f(xj), \pi))$
8. for all objective k where fk(xi) $<$ fk(xj) do set $\lambda^k = \lambda^k + \pi^k w$
9. end
10. if $\lambda = 0$ then set $\lambda$ to a randomly chosen vector from $\Lambda$
11. normalize($\lambda$)
12. find the solution yi which minimizes $\lambda$. f(xi) where yi $\in$ N(xi) and A(xi, yi) $\notin$ TLi
13. if TLi is full then remove oldest element from TLi
14. add A(yi, xi) to TLi as the newest element
15. set xi $=$ yi
16. if f(yi) is n − dominated by all point in ND then implement the point f(yi) into ND
and update $\pi$
17. if DRIFT − criterion is reached then set one randomly selected solution from
X equal to another randomly selected solution from X
18. set count $=$ count $+ 1$
19. end
20. until STOP − criterion is met
```

Figure 5: The basic MOTS procedure

In line 16, the new point is inserted into the ND-set if it is not dominated by it, and the points already defined in the ND-set that are dominated, if any, are deleted. If we wish, we can also save the solution itself. The equation of the range is defined as a function of the ranges of points in the set ND and can therefore be updated (of course they can only be calculated if we have at least two points defining a positive range in each goal). The use of point ranges in the ND set is a general suggestion in cases where no other knowledge of ranges is available. In this paper, we propose a random solution. It's a non-dominated border, and it's a non-dominated point. The next section will show how the dynamic dynamics of the drives in the X.

Finally, in line 18, the iteration counter is incremented by 1, and we are ready to continue with the next of the current solutions.

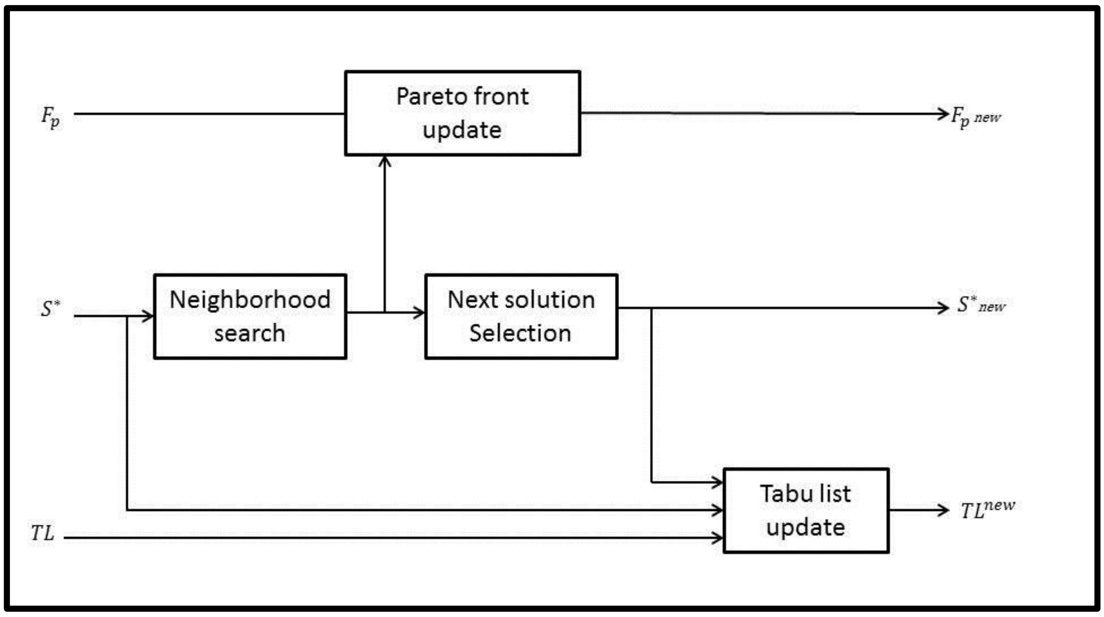

Figure 6: Structure of an iteration for a multi objective Tabu Search

6. NEIGHBORHOOD SEARCH

In the local search, we use three procedure. The new solution is a neighbor of the current solution ,the first is a swap procedure It is sufficient to select randomly from the current solution (in the solution space) two workers and then to swap between the jobs for which they are assigned (see Figure 7-a).

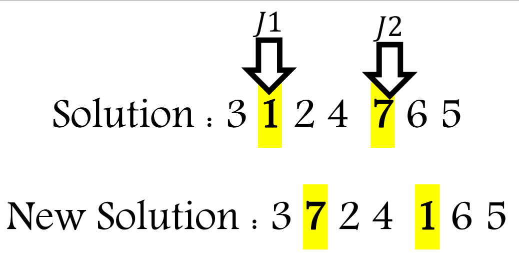

Figure (7-a):local search procedure (swap)

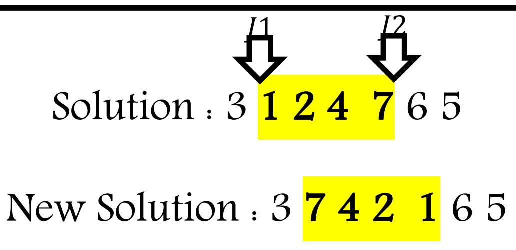

Figure (7-b):local search procedure (inersion)

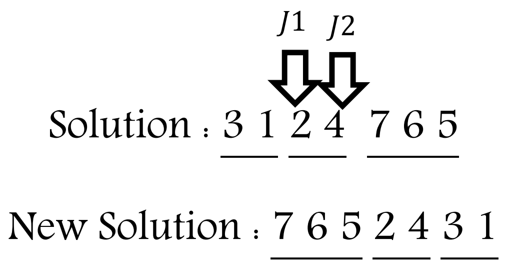

Figure (7-c):local search procedure (2-opt)

The second procedure we select randomly two points provided that they are second point is greater than the first and then we inversion vector between the two points (see Figure 7-b).

The third procedure we select randomly two points provided that they are second point is greater than the first and then We apply the following equation (see Figure 7-c):

$$S' = [s(j_2 + 1 : n) \quad s(j_1 : j_2) \quad s(1 : j_1 - 1)] \qquad if\ j_1 < j_2 \quad (4)$$

7. EXPERIMENTAL RESULTS

The algorithm was tested, using new data, on randomly chosen problems, with a range of coefficients of the objective functions in [0, 20]. Every problem is re-run ten times using a Core-i5 2.4 GHz PC with 4GB RAM in Windows operating system and programming using matlab 2013 a. It was compared to (TS,SA,GA.PSO) and we have evaluated their performances according to three measures of quality of E (approximate set of efficient solutions):

1. An average distance between $\hat{E}$ and E :

$$D_1(\hat{E}, \mathrm{E}) = \frac{\sum_{x \in E} d(\hat{E}, x)}{|E|} \qquad (5)$$

2. A worst case distance between $\hat{E}$ and E :

$$D_2(\hat{E}, \mathrm{E}) = \max_{x \in E} d(\hat{E}, x), \qquad (6)$$

3. A measure of the uniformity of quality of $\hat{E}$:

$$Ratio = \frac{D_2(\hat{E}, \mathrm{E})}{D_1(\hat{E}, \mathrm{E})} \qquad (7)$$

Table 1. The results of the comparison between algorithms for size problem between(15-60)

problem		TS				PSO				SA				GA			
number	size	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)
1	15	3.67	55.00	15.00	9.44	1.67	57.00	34.20	50.00	6.00	64.00	10.67	11.33	2.00	59.00	29.50	30.97
2	20	7.67	61.00	7.96	18.77	4.67	67.00	14.36	59.28	23.00	82.00	3.57	14.71	10.00	71.00	7.10	39.92
3	25	2.33	86.00	36.86	20.90	10.00	92.00	9.20	62.17	21.33	102.00	4.78	13.95	14.67	93.00	6.34	42.93
4	30	6.67	78.00	11.70	27.85	15.00	98.00	6.53	63.40	44.00	132.00	3.00	14.29	34.00	117.00	3.44	47.42
5	35	7.00	101.00	14.43	32.12	26.33	113.00	4.29	63.87	63.00	157.00	2.49	13.10	49.00	149.00	3.04	48.53
6	40	7.33	121.00	16.50	36.97	40.67	162.00	3.98	67.47	65.67	222.00	3.38	14.52	58.00	180.00	3.10	51.39
7	45	8.33	121.00	14.52	45.60	45.67	184.00	4.03	68.27	83.67	206.00	2.46	15.02	73.67	208.00	2.82	56.89
8	50	8.33	119.00	14.28	52.60	74.33	221.00	2.97	71.59	109.33	223.00	2.04	15.58	79.33	213.00	2.68	56.43
9	55	11.67	139.00	11.91	58.95	71.33	204.00	2.86	74.57	120.67	266.00	2.20	15.07	97.67	241.00	2.47	59.98
10	60	12.33	154.00	12.49	69.06	74.33	211.00	2.84	73.74	140.00	286.00	2.04	13.28	116.33	258.00	2.22	60.37

Table 2. The results of the comparison between algorithms for size problem between(75-120)

problem		TS				PSO				SA				GA			
number	size	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)
11	75	15.33	192.00	12.52	101.90	124.33	296.00	2.38	75.88	196.67	384.00	1.95	14.18	175.33	357.00	2.04	64.04
12	80	13.67	187.00	13.68	116.61	122.67	306.00	2.49	75.43	201.67	419.00	2.08	15.09	181.33	355.00	1.96	68.75
13	85	13.33	185.00	13.88	129.12	161.33	338.00	2.10	78.64	226.67	420.00	1.85	14.98	198.33	383.00	1.93	68.78

14	90	17.00	202.00	11.88	139.39	146.33	342.00	2.34	79.36	230.67	421.00	1.83	16.65	215.33	394.00	1.83	67.45
15	95	16.00	207.00	12.94	153.36	204.67	394.00	1.93	80.81	261.33	471.00	1.80	16.47	239.00	466.00	1.95	69.75
16	100	17.00	221.00	13.00	171.07	196.00	412.00	2.10	85.07	280.67	490.00	1.75	16.91	257.67	506.00	1.96	78.13
17	105	21.00	215.00	10.24	178.29	267.00	471.00	1.76	86.30	304.00	516.00	1.70	16.43	283.67	519.00	1.83	77.16
18	110	14.67	231.00	15.75	192.68	194.33	414.00	2.13	82.27	313.00	535.00	1.71	15.24	294.33	519.00	1.76	76.72
19	115	17.33	257.00	14.83	207.68	320.67	564.00	1.76	83.59	326.33	563.00	1.73	15.40	306.33	548.00	1.79	78.34
20	120	19.00	249.00	13.11	211.93	245.00	481.00	1.96	84.05	355.33	593.00	1.67	15.43	316.00	542.00	1.72	81.71

Table 3. The results of the comparison between algorithms for size problem between(150-250)

problem		TS				PSO				SA				GA			
number	size	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)	D1	D2	Ratio	Time(s)
21	150	42.00	310.00	7.38	44.16	326.33	621.00	1.90	63.10	460.67	765.00	1.66	9.87	427.33	733.00	1.72	183.47
22	160	55.00	358.00	6.51	22.34	335.67	617.00	1.84	52.43	497.00	807.00	1.62	22.32	469.67	778.00	1.66	95.88
23	170	50.67	360.00	7.11	21.47	521.67	826.00	1.58	63.87	540.00	906.00	1.68	10.41	502.33	817.00	1.63	95.93
24	180	42.33	338.00	7.98	40.01	585.33	901.00	1.54	60.04	587.33	948.00	1.61	10.50	563.00	859.00	1.53	88.30
25	190	57.33	377.00	6.58	38.59	617.67	948.00	1.53	57.44	622.00	963.00	1.55	11.31	569.67	891.00	1.56	65.66
26	200	46.33	397.00	8.57	30.09	547.67	906.00	1.65	63.45	648.33	980.00	1.51	11.86	623.00	980.00	1.57	77.36
27	210	56.00	423.00	7.55	50.59	683.33	1056.00	1.55	68.38	707.67	1070.00	1.51	12.31	663.33	1040.00	1.57	139.46
28	220	53.00	410.00	7.74	63.20	670.00	1067.00	1.59	79.69	748.67	1121.00	1.50	21.65	705.33	1071.00	1.52	114.77
29	230	51.33	449.00	8.75	68.03	768.67	1149.00	1.49	65.47	768.00	1159.00	1.51	11.09	745.33	1164.00	1.56	186.98
30	240	48.67	440.00	9.04	91.37	824.00	1234.00	1.50	60.75	822.00	1244.00	1.51	23.24	788.33	1208.00	1.53	77.88
31	250	65.33	471.00	7.21	66.47	864.67	1258.00	1.45	121.73	874.00	1336.00	1.53	13.88	813.33	1279.00	1.57	163.35

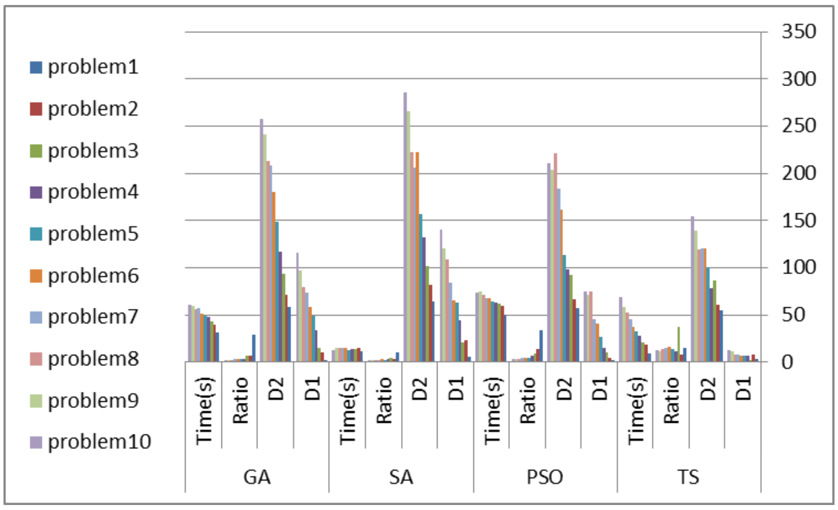

Figure (8-a): the comparison between algorithms (problem1-problem10)

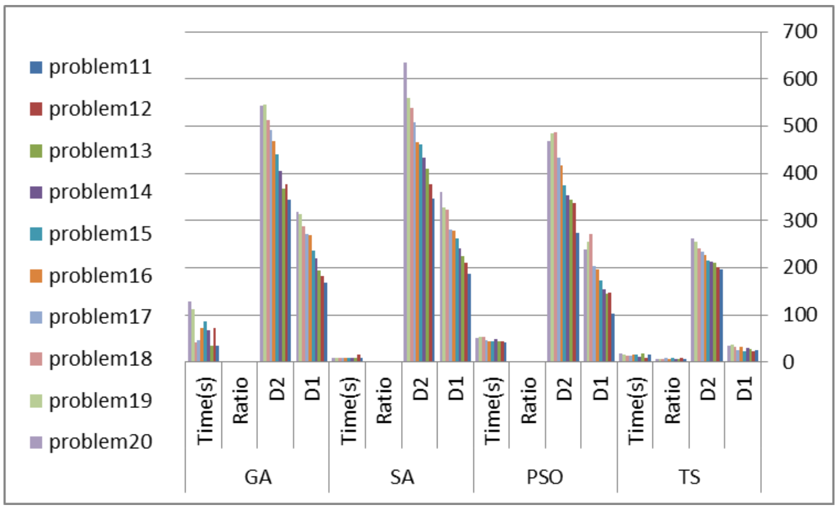

Figure (8-b): the comparison between algorithms (problem11-problem20)

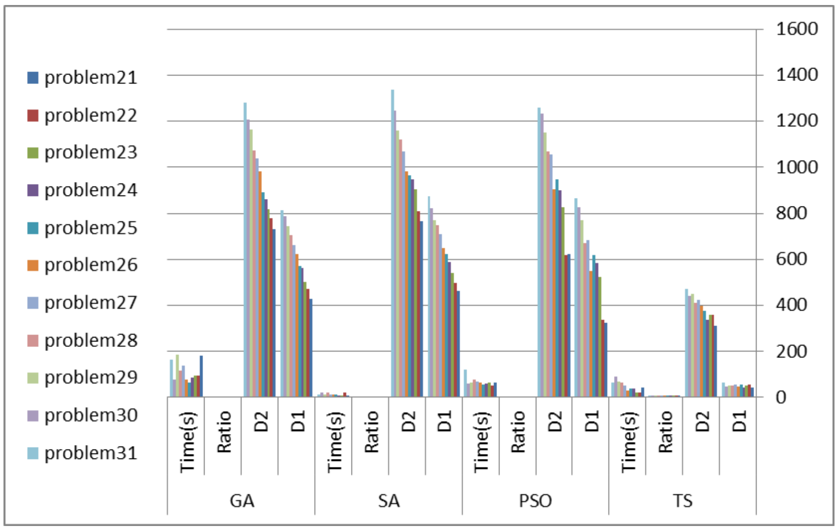

Figure (8-c): the comparison between algorithms (problem21-problem31)

8. DISCUSSION RESULT

In Tables 1,2 and 3, we provide a summary of the results obtained by the TS Algorithm And PSO,SA and GA Algorithms. By comparing the TS with three Algorithms, it can be seen that In terms of the quality of the solutions, the TS goes beyond the determination of PSO More than 1 and 2 effective solutions for at the same time. For the other cases, the TS provides all the advantages Solutions.

9. CONCLUSION

In this article, a Tabu search algorithm using three local moves (2-opt and inversion and swap) has been proposed to solve a multi-objective assignment problem with three or more objectives. The first results are encouraging. Indeed, when compared to Pso, SA and GA, the Tabu search algorithm allowed for much better solutions for the same time. In the future work, many computational experiments could be studied for a multi-objective assignment problem and other classes of multi-objective combinatorial optimization problems. In addition, we propose to parallelize the method: instead of choosing a new current solution x^r from the set of efficient local solutions, in the neighborhood of x^r, we reiterate the method with all local solutions efficient to get a better Quality solutions.

REFERENCE

[1] Day, Richard O., and Gary B. Lamont. "Multiobjective quadratic assignment problem solved by an explicit building block search algorithm–MOMGA-IIa." European Conference on Evolutionary Computation in Combinatorial Optimization. Springer Berlin Heidelberg, 2005.

[2] Serafini, Paolo. "Some considerations about computational complexity for multi objective combinatorial problems." Recent advances and historical development of vector optimization. Springer Berlin Heidelberg, 1987. 222-232.

[3] Ge, Yue, Minghao Chen, and Hiroaki Ishii. "Bi-criteria bottleneck assignment problem." Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American. IEEE, 2012.

[4] Blum, Christian, and Andrea Roli. "Hybrid metaheuristics: an introduction." Hybrid Metaheuristics. Springer Berlin Heidelberg, 2008. 1-30.

[5] Adiche, Chahrazad, and Méziane Aïder. "A Hybrid Method for Solving the Multi-objective Assignment Problem." Journal of Mathematical Modelling and Algorithms 9.2 (2010): 149-164.

[6] Przybylski, Anthony, Xavier Gandibleux, and Matthias Ehrgott. "Two phase algorithms for the bi-objective assignment problem." European Journal of Operational Research 185.2 (2008): 509-533.

[7] Ulungu, Ekunda Lukata, and Jacques Teghem. "Multi- objective combinatorial optimization problems: A survey." Journal of Multi- Criteria Decision Analysis 3.2 (1994): 83-104.

[8] Blum, Christian, and Andrea Roli. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison." ACM Computing Surveys (CSUR) 35.3 (2003): 268-308.

[9] Coello, Carlos A. Coello, Gary B. Lamont, and David A. Van

Veldhuizen. Evolutionary algorithms for solving multi-objective problems. Vol. 5. New York: Springer, 2007.

[10] Balicki, Jerzy. "Tabu programming for multiobjective optimization problems." International Journal of Computer Science and Network Security 7.10 (2007): 44-51.

[11] Hansen, Michael Pilegaard. "Tabu search for multiobjective optimization: MOTS." Proceedings of the 13th International Conference on Multiple Criteria Decision Making. 1997.

[12] Jaffres-Runser, Katia, Jean-Marie Gorce, and Cristina Comaniciu. "A multiobjective Tabu framework for the optimization and evaluation of wireless systems." arXiv preprint arXiv:0907.3777 (2009).

[13] Deb, Kalyanmoy, Karthik Sindhya, and Jussi Hakanen. "Multi-objective optimization." Decision Sciences: Theory and Practice. CRC Press, 2016. 145-184.

[14] Salehi, Kayvan. "An approach for solving multi-objective assignment problem with interval parameters." Management Science Letters 4.9 (2014): 2155-2160.