SOLVING MULTI OBJECTIVE ASSIGNMENT PROBLEM USING TABU SEARCH ALGORITHM

A. M. Kadhem
Department of Statistics, College of Economic & Administration,University of Baghdad, Baghdad, Iraq
Contact: **Munaam53@yahoo.com**

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 uses heuristically to create non-dominated alternatives to multi-objective combinatorial optimization problems. MOTS works with a set of current solutions that appreciate the manipulation of weights, which is optimized towards the non-dominated border while trying to disperse on the border. It generate some problem to measure the effectiveness of the algorithm in three different objective sizes of the question and compare the results with the simulated annealing algorithm, the genetic algorithm and particle swarm optimization by three measurements. This algorithm has displayed 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 search for one to one identity amongst n jobs and n workers, the main idea is the reduction of the total cost or maximizing the total efficiency of the assignments. The classical AP has widely been studied in literatures and several algorithms are accessible for solving it, for example, the Hungarian method [5], the flow type method [3], the Munkres method [9].

In real world most of the assignment problem possess multiple objective such as reduction in cost and simultaneously increasing difficult problems, number of objectives and switching from the problem of multi-algorithm solution of the border into a more complex geometry problem as seen in most issues that multiple objective does not have a polynomial algorithm and also in the field of multi-objective combinatorial optimization problem.

2. The Multi objective Optimization Problem

The problem of multi-objective optimization can be explained as a problem of discovering a vector of decision variables that meet the constraints and optimizing a vector function whose objective functions is represented by elements. These functions are mathematical explanation for performance conditions which are always in conflict with each other. So, the term "optimize" means discovering such solution that would provide the values of all objective functions which would be accepted by the decision maker.

Multi-objective optimization's general problem can now be correctly 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) for particular set $x_1^*, x_2^*, \dots, x_n^*$ which provides the values for optimal all-purpose functions. The constraints given by (1) and (2) define the feasible region n and any

A workable solution is define by the point $\vec{x}$ in Ω. 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 non-commensurable conditions that must be put into consideration. The constraints $g_i(\vec{x})$ and $h_i(\vec{x})$ represent the restraints enforced on the decision variables. The vector i* is reserved to denote the optimal solutions (there are commonly more than one).

3. Multi objective Assignment problem

The expression of general multi-objective combinatorial optimization problem is as follows:

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

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

Importantly in (MOCO), the term "min" is presented in a quotation marks because, there is no single solution that exist that is minimal on all objectives. As a result, numerous concepts must be proven to explain what an optimal solution is. The most frequently used one is the dominance relation (also known as Pareto dominance) (Fig. 1):

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

unimodularity. The Hungarian method or the successive shortest paths method (Papadimitriou and Steiglitz, 1982; Ahuja et al., 1993) are the well-known efficient algorithm for solving it. In this paper, we are considering the assignment problem with two objectives (BAP). This can be formulated as described below:

This solution can be detected via numerous methods, where the solution is given by a permutation vector φ, of n elements:

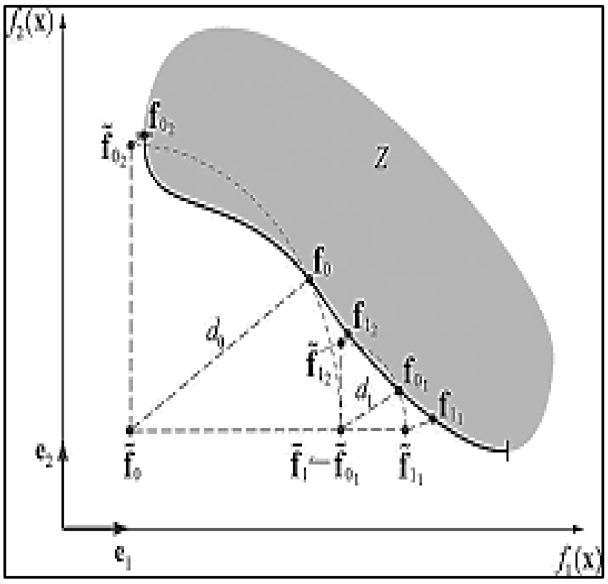

Fig. 1 Dominations in the Pareto sense in a bi-objective Space

trajectories that cannot be divided into these two phases. The features of the trajectory give information on the behavior of the algorithm and how effective it is with regards to the case addressed. Therefore, it should be underscored that the dynamics is the result of the summation of the algorithm, the problems representation and the instance of the problem. In fact, the problem representation and the neighborhood structure define the research landscape. The algorithm provides the description of the strategy employed for exploring the landscape and finally the actual features of the search space are defined by the instance of the problem to be solved. Firstly, we will describe the basic local search algorithms prior to moving on to more complex strategies and lastly we deal with general exploration strategy algorithms which can integrate other trajectory methods as components.

4.1 Basic Local Search: Iterative Improvement

Basic local search is generally referred to as iterative enhancement, since every movement is only carried out if the resulting solution is better than the current solution. The algorithm stops immediately, it finds a local minimum. The high level algorithm is defined in Figure 2. The Enhance (N (s)) function can either be at the extreme or at a first improvement, or at any intermediate option. The first analyzes the N (s) of the neighborhood and select the first

A better solution than s is that the latter exhaustively explores the neighborhood and returns one of the solutions with the lowest objective function value. Both methods terminate at local minimums. Therefore, their performance strongly rely on the definition of S, f and N. The performance of Iterative improvement procedures on COP is mostly unsatisfactory. Therefore, numerous techniques have been developed to stop algorithms from being trapped

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

```
Procedure basic LS
s ← GenerateInitialSolution()
   repeat
 s ← Improve(N(s))
until no improvement is possible
```

Fig. 2. Algorithm: Iterative Improvement.

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

The difficulty of MOCO problems was generated from the following two factors.

• The resolution of a MOCO problem needs thorough cooperation with the decision makers (DM), which transforms into a specific high requirements for the effective tools utilized in generating effective solutions.

• Several combinatorial problems are challenging, even in single-objective versions while their multiple objective versions are more difficult.

Furthermore, several single-purpose combinatorial difficulties belong to the group of NP-hard problems. Creating effective solutions in a MOCO problem is apparently not easier than searching solutions that optimize specific objectives and in most cases is more difficult. For instance,

The tools utilize for generating effective solutions in MOCO, like single-purpose optimization methods, can be grouped into one of the following:

• exact procedures;
• specialized heuristic procedures;
• meta-theorist procedures.

4. Local Search algorithms

In this segment, we give an account of the metaheurists called trajectory methods. The trajectory methods term is used due to the searching process carried out by these methods which is characterized by a trajectory in the search space. Subsequently, a successive solution might or might not be in the locality of the present solution. The process of looking for trajectory methods can be considered as the (discrete) evolution of a discrete dynamic system (Bar-Yam 1997]. The algorithm starts from a preliminary state (the initial solution) describing the trajectory in the state space. The system dynamics depend on the strategy used; Simple algorithms create a trajectory which is made of two parts: a transitory phase followed by an attractor (a fixed point, a cycle or a complex attractor). Algorithms with advanced strategies produce more complex

Definition 3 A solution x∗ ∈ S is regarded as (Pareto) efficient for (MOCO) if and only it does not exist in any other feasible solution $x \in S$, in such that $f_i(x) \leq f_i(x^*)$ for $i = 1..p$ with at least one $j \in \{1,\dots,p\}$ 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 known as the Pareto optimal set, is always denoted by E and the image of E in Z is called the non-dominated frontier or the Pareto optimal front which is denoted byZ_E.

The single objective assignment problem (AP) is an integer programming problem which can be solved using a linear program because of the constraint matrix in total means that the algorithm is the result of the combination of two strategies: random walk and iterative improvement. In the first phase of this research, the bias of the improvements is minimal and permits the exploration of the research space; this erratic component reduces gradually, which results to search for converge to a minimum (local). The probability of accepting uphill movements is regulated by two factors: the difference among 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] indicated that, under specific conditions on the cooling schedule, the algorithm converges in probability to a global one.

The intensity of water can be reduced with water cooling. For example, at the beginning of the search, T could be constant or reduce linearly for the sampling of the search space; Then T could go along the geometry rule, to converge to the minimum at the end of the search.

Most successful variants are non-monotonic cooling schedules as described by Osman[19] and Lundy and Mees 1986 [20],. Non-monotonic cooling schedules are categorized by alternating the cooling and reheating phases, which offer an oscillating balance between diversification and intensification.

The cooling system and the initial temperature should adapt to the particular problem because the structure of the research landscape determines the cost of exhausting local minima. The easiest way to empirically determine the starting temperature T_0 is to first sample the search space with a random walk to approximate the average and the variance of the values of the objective function and also implementation of more elaborate schemes can be performed [21].

4.3 Tabu search

Tabu Search (TS) is one of the most common cited metaheuristics used for CO problems. Glover was first to introduce the basic ideas of TS based on former ideas [22][23].

The method and its concepts can be found in Glover and Laguna [24]. TS clearly uses the search history, for escaping local minima and for the implementation of an exploration strategy.

A simple version of TS would be first described to present the basic concepts. Further, we will explain a more applicable algorithm and lastly we will discuss some improvements.

in local minima, which is performed by the addition of mechanisms that permit them to escape local minima. This also infers that the terminating conditions of the metaheurstic algorithms are more complex than simply reaching a local minimum. Certainly, 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 usually regarded as the oldest among the metaheurstic and definitely one of the premier algorithms that possess an explicit strategy for escaping the local minima. The statistical mechanics are the origins of the algorithm (Metropolis algorithm) and it was presented for COP in Kirkpatrick et al. [25] and Cerny [26] as a search algorithm. The ultimate idea is to permit movements leading to quality solutions worse than the current solution (uphill movements) in order to escape the local minima. The probability of such a movement is reduced during the search. The high level algorithm is described in Figure 3.

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

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

The algorithm starts by creating 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 commonly computed following the Boltzmann distribution $e^{(-\frac{f\,(s\prime) - f\,(s)}{T})}$.

There is a decrease in temperature T during the search process at the beginning of the search, the probability of accepting uphill movements is higher and it steadily decreases, congregating towards a simple iterative improvement algorithm. This process is similar to the metal and glass process of annealing, which gives a low configuration of energy when cooled with a suitable cooling program. With regards to the search process, it corresponding tabu lists describe the tabu conditions that are used for filtering the neighborhood of a solution and for creating the permitted set. Storing attributes instead of complete solutions has greater efficient, but it results to loss of information because forbidding an attribute implies affecting the state of the table to probably more than one solution. Consequently, it is promising that non-visited solutions of good quality are left out from the permitted play. To circumvent this problem,

Suction criteria are defined to allow a solution to be included in the permitted game, even though it is prohibited by the tabu conditions. The Suction Criteria set the suction settings for generating the allowed set. The most widely utilized aspiration criterion selects better when compare to the best solutions. The complete algorithm, as defined above, is reported in Figure 4. Taboo lists are only one way to take advantage of the search history. They are usually recognized using the short-term memory. The information collected during the search procedure can also be very valuable, particularly for a strategic orientation of the algorithm. This particular long-term memory is commonly added to TS by referring to four principles i.e. recurrence, frequency, quality and influence. The memory records based on recurrence for each solution (or feature) which is the most recent iteration that was involved. The memory based on the orthogonal frequency keeps record of how frequent each solution (attribute) was visited. This information identifies the regions (or Subsets) of the solution space where the search was restricted or where there is a remained of a large number of iterations. This kind of information regarding the past is generally exploited for research diversification. The third principle (that is, quality) refers to the accumulation and retrieval of information from the search history for identifying good solution components. This information can be incorporated efficiently into the construction of the solution. Other metaheuristics (for example, Ant Colony Optimization) explicitly utilize this principle to distinguish the right combinations of components in the solution. Lastly, influence is a property concerning the choices made during the search and can be used for indicating which choices are the most vital. Overall, the TS field is a rich source of ideas. Several 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 [27], Tabu reactivity to the MAXSAT problem [29], and assignment problems]. TS approaches dominate the challenging area of Job Shop Scheduling (JSS) and the vehicle routing area (VR).

Multi objective Tabu search algorithms

The multi-objective tabu search procedure, MOTS, functions using a set of current solutions that are concurrently optimized against the non-dominated border. The points of the existing solutions are sought to cover the entire boundary and, for each solution in numerous times. An optimization direction is produce so that it tends to migrate 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 maintains its own tabu list. In the following, we will annotate the Pascal contour of the MOTS database in FIG. 5.

```
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 put on a better local search for improvement as the elementary ingredient and utilizes a short-term memory for escaping local minima and for cycle's avoidance. The short-term memory is implemented in the form of list of tables which permits the tracking of the last solutions visited and stop the displacements towards them. The area of the current solution is so limited to solutions that do not belong to the list of tabuses. This set will be refer to as defined in the following,

At each iteration, the best solution of the authorized game is chosen as the new current solution. Likewise, 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 [34]. 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. Application of a tabu list inhibits the 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 other hand, a huge regime of taboos forces the research process to discover larger regions because it prevents revisiting a greater number of solutions. The duration of the tabu can differs during the search, leading to more robust algorithms. [27] described the model for this, in which the tabu regime is occasionally reset indiscriminately from the interval [lmin, lmax]. A more advanced use of a dynamic taboo regime is described in [28,29],

The rate of pension increased if evidence for repetition of solutions exist (Hence greater diversification is required), Eight strategies to elude stopping the search when the allowed play is empty include choosing the less solution recently visited, even if it is a tabu. Nine cycles of higher period are possible, since the list of tabu has a finite length l which is lesser than the Cardinality of the search space. While it drops if no improvements is observed (therefore the strength should be intensify). More advanced ways for creating a dynamic taboo regime are defined in Glover [30]. Nevertheless, the implementation of short-term memory as a list having total solutions is not practical since managing a list of solutions is very ineffective. So, instead of the solutions themselves, the solution characteristics are stored. Attributes are typically solution components, movements, or differences among two solutions. As more than a single feature can be considered, a tabu list is entered for each characteristic. The set of features and the distance function (d) as a function of certain measurements in the objective function space using the weights of the range. The influence is produce 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. This 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.

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 commence the loop which continually vacate 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. Thus, each element of the weight vector is defined as a function of the proximity of other points for this purpose. Though, 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

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

a common 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 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 increase by 1, and we are ready to continue with the next of the current solutions.

In line 16, the new point is inserted into the ND-set if it is not dominated by it, and the points previously defined in the ND-set that are dominated if any is deleted and we can also save the solution if interested. The equation of the range is defined as a function of the ranges of points in the set ND that can 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

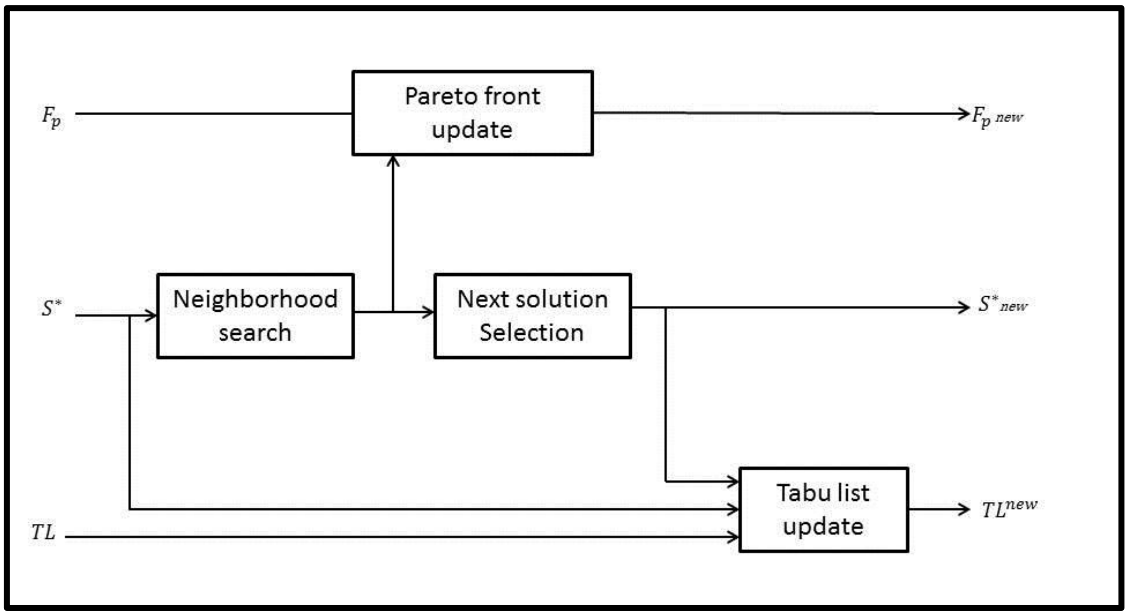

Figure 6: Structure of an iteration for a multi objective Tabu Search

5. Neighborhood search

In the local search, three procedure were used. The novel solution is a neighbor of the existing solution. The first is a swap procedure which is sufficient for selecting randomly from the existing solution of (in the solution space) two workers and for swapping between the jobs for which they were assigned (see Figure 7-a).

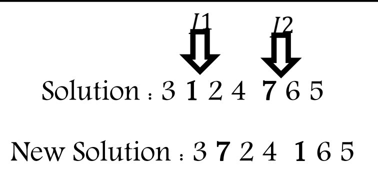

Figure (7-a):local search procedure (swap)

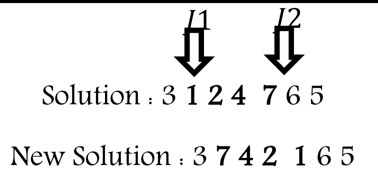

Figure (7-b):local search procedure (inersion)

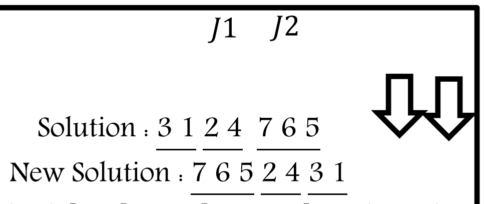

Figure (7-c):local search procedure (2-opt)

In the second process, we randomly select two points provided that the second point is greater than the first and with inversion vector between the two points (see Figure 7-b). The third process we randomly select two points provided their second point is greater than the first and then the following equation is applied (see Figure 7-c):

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

6. Experimental Results

The algorithm was tested with new data on problems that were randomly chosen with a range of coefficients of the objective functions in [0, 20]. Each 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 then compared to (TS,SA,GA.PSO) and we 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

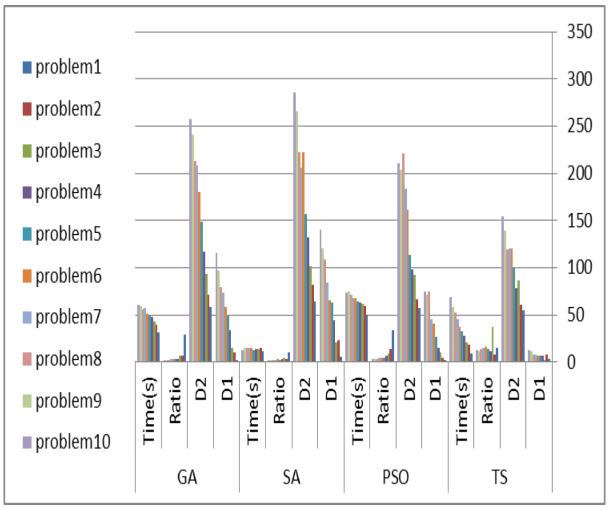

Figure (8-a): the comparison between algorithms (problem1-problem10)

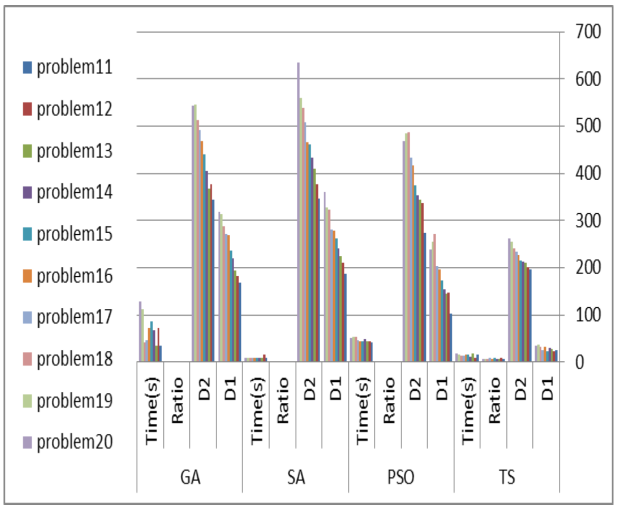

Figure (8-b): the comparison between algorithms (problem11-problem20)

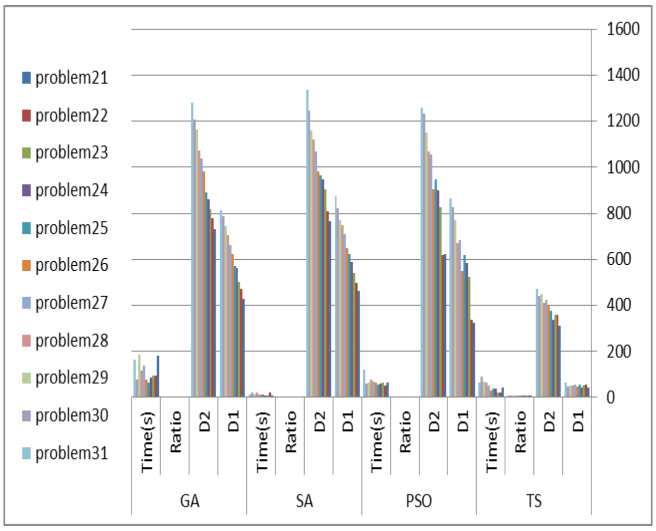

Figure (8-c): the comparison between algorithms (problem21-problem31)

DISCUSSION RESULT

In Tables 1, 2 and 3, we make available a summary of the results acquired by the TS Algorithm and PSO, SA and GA Algorithms. Comparing the TS with three Algorithms, it was observed that in terms of the quality of the solutions the TS goes beyond the determination of PSO more than 1 and 2 effective solutions at the same time. For other cases the TS provided all the advantages Solutions.

7. CONCLUSION

In this article, a Tabu search algorithm using three local moves (2-opt, inversion and swap) has been proposed to solve a multi-objective assignment problem with three or more objectives. The first results are promising. When compared to Pso, SA and GA, the Tabu search algorithm permit much better solutions for 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. Finally, 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. L. R. Ford Jr. and D. R. Fulkerson, Flows in Networks. Princeton
University Press, New Jersey, 1962.
4. 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.
5. H. W. Kuhn, The Hungarian method for th assignment problem. Naval Logistics Quarterly 2, 83-97, 1955.
6. Blum, Christian, and Andrea Roli. "Hybrid metaheuristics: an introduction." Hybrid Metaheuristics. Springer Berlin Heidelberg, 2008. 1-30.
7. 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.
8. 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.
9. J. Munkres, Algorithms for the assignment and transportation problems. Journal of Society for Industrial and Applied Mathematics 5(1), 32-38, 1957.

10. Ulungu, Ekunda Lukata, and Jacques Teghem. "Multi- objective combinatorial optimization problems: A survey." Journal of Multi- Criteria Decision Analysis 3.2 (1994): 83-104.
11. Blum, Christian, and Andrea Roli. "Metaheuristics in combinatorial optimization: Overview and conceptual comparison." ACM Computing Surveys (CSUR) 35.3 (2003): 268-308.
12. 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.
13. Balicki, Jerzy. "Tabu programming for multiobjective optimization problems." International Journal of Computer Science and Network Security 7.10 (2007): 44-51.
14. Hansen, Michael Pilegaard. "Tabu search for multiobjective optimization: MOTS." Proceedings of the 13th International Conference on Multiple Criteria Decision Making. 1997.
15. 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).
16. Deb, Kalyanmoy, Karthik Sindhya, and Jussi Hakanen. "Multi-objective optimization." Decision Sciences: Theory and Practice. CRC Press, 2016. 145-184.
17. Salehi, Kayvan. "An approach for solving multi-objective assignment problem with interval parameters." Management Science Letters 4.9 (2014): 2155-2160.
18. Aarts, Emile HL, and Jan Karel Lenstra, eds. Local search in combinatorial optimization. Princeton University Press, 1997.
19. Osman, Ibrahim Hassan. "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem." Annals of operations research 41.4 (1993):421-451
20. Lundy, Miranda, and Alistair Mees. "Convergence of an annealing algorithm." Mathematical programming 34.1 (1986): 111-124.
21. Ingber, Lester. "Statistical mechanics of nonlinear nonequilibrium financial markets: Applications to optimized trading." Mathematical and computer modelling 23.7 (1996): 101-121.
22. Glover, Fred. "Future paths for integer programming and links to artificial intelligence." Computers & operations research 13.5 (1986): 533-549.
23. Glover, Fred. "Heuristics for integer programming using surrogate constraints." Decision Sciences 8.1 (1977): 156-166.
24. Glover, Fred, and Manuel Laguna. "General purpose heuristics for integer programming—part II." Journal of Heuristics 3.2 (1997): 161-179.
25. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.
26. Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications, 45(1), 41-51.
27. Taillard, Éric. "Robust taboo search for the quadratic assignment problem." Parallel computing 17.4-5 (1991): 443-455.
28. Battiti, Roberto, and Marco Protasi. "Reactive search, a history-sensitive heuristic for MAX-SAT." Journal of Experimental Algorithmics (JEA) 2 (1997): 2.
29. Battiti, R., & Tecchiolli, G. (1994). The reactive tabu search. ORSA journal on computing, 6(2), 126-140.
30. Nowicki, Eugeniusz, and Czeslaw Smutnicki. "A fast taboo search algorithm for the job shop problem." Management science 42.6 (1996): 797-813.
31. Gendreau, M., Laporte, G., & Potvin, J. Y. (2001, January). Metaheuristics for the capacitated VRP. In The vehicle routing problem (pp. 129-154). Society for Industrial and Applied Mathematics
32. Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). Network flows: theory, algorithms, and applications.
33. Bar-Yam, Y. (1997). *Dynamics of complex systems* (Vol. 213). Reading, MA: Addison-Wesley.
34. Stützle, T., & Hoos, H. (1999). The max-min ant system and local search for combinatorial optimization problems. In *Meta-heuristics*(pp. 313-329). Springer US.