4. Наумейко, И.В. Еще одна динамическая модель марковской системы человек-машина-среда, на которую действуют вредные факторы [Текст]/ И.В. Наумейко, Р.Дж. Аль-Азави // Харьков, Радиотехника 2013 (в печати).
5. Al-Azawi, R. J. A dynamic model of Markovian Human-Machine-Environment system that is effected by some hazard [Текст]/ R. J. Al-Azawi //Инновационный потенциал украинской науки - XXI век, Харьков апреля 2013 г, , вып. 20 (в печати).
6. Севастьянов, Б.А. Формулы Эрланга в телефонии при произвольном законе распределения длительности разговора [Текст]/ Б.А. Севастьянов// Труды III Всесоюзного математического съезда. Т.IV М.: АН СССР, 1959.
7. Хинчин, А. Я. Работы по математической теории массового обслуживания [Текст]/ А. Я. Хинчин// Под редакцией Б. В. Гнеденко. – М.: Физматгиз, 1963, 236 с.
8. Ивченко, Г.И. Теория массового обслуживания [Текст]/ Г.И. Ивченко, В.А. Каштанов , И.Н. Коваленко // Высшая школа, 1982, 256 с.
9. Беккенбах, Э. Неравенства [Текст]/ Э. Беккенбах, Р. Беллман. // Мир, 1965.
10. Самойленко, А.М. Дифференциальные уравнения [Текст]/ А.М. Самойленко //Высшая школа, 1989, 383 с.

УДК 519.87

FUZZY DISTANCES AND THEIR APPLICATIONS ON FUZZY SCHEDULING

Hanan A. Cheachan*
E-mail: hanan_altaai@yahoo.com
Hussam A.A. Mohammed
Department of Mathematics
College of Education for Pure Sciences
University of Karbala
Iraq - Karbala, 56001
E-mail: hussammath@yahoo.com
Faria A. Cheachan*
E-mail: faria_altaai@yahoo.com
Department of Mathematics
Univ. Sciences
University of Mustansiriya
Iraq - Bogdad, Almustansiriya, 46007

Мета даної роботи полягає в тому, щоб розробити нечітку проблему планування для того, щоб вирішити багатоцільові функції на єдиних машинних проблемах планування, коли тривалість обробки і число закінчення терміну - трикутні нечіткі числа. Ми використовуємо нечіткі поняття функції відстані, які введені за теоремою Лам і Цай

Ключові слова: нечітка проблема планування, машинне планування, методи локального пошуку, заборонений пошук

Цель данной работы состоит в том, чтобы разработать нечеткую проблему планирования для того, чтобы решить многоцелевые функции на единственных машинных проблемах планирования, когда продолжительность обработки и число истечения срока - треугольные нечеткие числа. Мы используем нечеткие понятия функции расстояния, которые введены по теореме Лам и Цай

Ключевые слова: нечеткая проблема планирования, машинное планирование, методы локального поиска, запрещенный поиск

1. Introduction

In most of cases in our life, the data obtained for decision making are only approximately known. In 1965, Zadeh [18] introduced the concept of fuzzy set theory to meet that problem. In 1978, Dubois and Prade [4] defined the fuzzy numbers as a fuzzy subset of the real line. In 1991, Kaufmann and Gupta [12], considered a distance measure of two fuzzy number combined by the interval of of fuzzy numbers. In 1997, Heilpern [9] proposed three definitions of the distance between two fuzzy numbers. Lam and Cai [13] gave a fuzzy function from measuring the distance between fuzzy number and also showed by experiments their distance function given very good approximation to the expected distance in numerous situations. The singe machine case is of great importance since there are some general problems of this type which can be solved in polynomial time. The assumption that all parameters are determined restricts the practical aspect of scheduling since, for many real-world processes the exact values of parameters are not known advance the natural approach to modeling the uncertainty is a stochastic one in which the parameters are given as random variables. Unfortunately, such an approach leads to difficult problems from the computational point of view and only some special cases can be effectively solved. The alternative approach to modeling imprecision is a fuzzy one in which the parameters are given in the form of fuzzy numbers. This approach turns out be easier than stochastic one and there are some general problems, which can be solved in polynomial time. In the recent decade there have appeared some papers dedicated to fuzzy single scheduling ([2], [17]). The first aim of this paper is to propose a general formulation of multi-objective function with fuzzy due dates and fuzzy processing time. The second aim we will apply some local search methods (Threshold accepted (TA), Tabu search (TS), and Memetic algorithm (MA)).

2. Preliminaries

In this section we specify the context of this study and recall basic definitions that will be used in the following. We also present the principle underlying the main approaches for defining fuzzy distances.

2.1. Definition [7]

Let X be a nonempty set of points. A fuzzy subset $\tilde{A}$ of X is a function $\tilde{A}: X \rightarrow [0,1]$ for each $x \in X$, $\tilde{A}(x)$ is called the degree of membership of x in $\tilde{A}$.

2.2. Definition [16]

A fuzzy number $\tilde{A}$ is a fuzzy subset of the real numbers R which is denied, in general, as follows:

$$\tilde{A}(x)=\begin{cases} 0 & \text{for } x<a \\ f_{\tilde{A}}(x) & \text{for } a \le x<c \\ 1 & \text{for } c \le x \le d \\ g_{\tilde{A}}(x) & \text{for } d<x \le b \\ 0 & \text{for } b<x \end{cases}$$

where $f_{\tilde{A}}$ and $g_{\tilde{A}}$ are respectively, non-decreasing, and non-increasing functions.

2.3. Definition [7]

Let $\tilde{A}$ be a fuzzy number of X, the $\alpha-\text{cut}$ of $\tilde{A}$ denoted $\tilde{A}_\alpha$ is defined by $\tilde{A}_\alpha = \{x \in X : \tilde{A}(x) \ge \alpha\}$ where $\alpha \in [0,1]$. On the other hand, $0-\text{cut}$ of $\tilde{A}$ is called the support of $\tilde{A}$ and denoted by supp $\tilde{A}$. It follows from the properties of the membership function of a fuzzy number $\tilde{A}$ that each its $\alpha-\text{cut}\ \tilde{A}, \alpha \in [0,1]$, is a closed interval. We will denote it by $\tilde{A}_\alpha = [\underline{a}_\alpha, \overline{a}_\alpha]$.

2.4 Definition [16]

A fuzzy number $\tilde{A}$ in R is a triangular fuzzy number if its membership function $\tilde{A}: R \rightarrow [0,1]$ is equal to:

$$\tilde{A}=\begin{cases} 0 & \text{for } x<a \\ \dfrac{x-a}{b-a} & \text{for } a \le x<b \\ \dfrac{c-x}{c-b} & \text{for } b \le x<c \\ 0 & \text{for } c \le x \end{cases},$$

with $a \le b \le c$. The triangular fuzzy number can be denoted by $\tilde{A}=(a,b,c)$.

2.5. Operations of Triangular Fuzzy Numbers [16]

The fuzzy arithmetic operations of triangular fuzzy numbers are described as follows. If two triangular fuzzy numbers are $\tilde{A}=(a_1,a_2,a_3)$ and $\tilde{B}=(b_1,b_2,b_3)$:

(1) Addition:-
$\tilde{A}+\tilde{B}=(a_1,a_2,a_3)+(b_1,b_2,b_3)=(a_1+b_1,a_2+b_2,a_3+b_3)$.

(2) Subtraction:-
$\tilde{A}-\tilde{B}=(a_1,a_2,a_3)-(b_1,b_2,b_3)=(a_1+b_3,a_2-b_2,a_3+b_1)$.

(3) Multiplication:-
$\tilde{A}.\tilde{B}=(a_1,a_2,a_3).(b_1,b_2,b_3)=(a_2b_1+b_2a_1,a_2b_2,a_2b_3+b_2a_3)$.

(4) Division:-
$$\tilde{A}/\tilde{B}=(a_1,a_2,a_3)/(b_1,b_2,b_3)=\left(a_2/b_1+a_1/b_2,\ a_2/b_2,\ a_2/b_3+a_3/b_2\right).$$

3. Model Development

In 2012, Hussam and et. al. [10] gave the formal of cost function $\tilde{L}_{\max}(\tilde{C}_j,\tilde{D}_j)$ with fuzzy completion time and fuzzy due date, in this section we give some details of derive $\tilde{L}_{\max}(\tilde{C}_j,\tilde{D}_j)$ and find the final formula of $\tilde{L}_{\max}(\tilde{C}_j,\tilde{D}_j)$ which is needed later. First, we use the following function for measuring the distance between fuzzy numbers [14].

$$\tilde{d}(\tilde{A},\tilde{B})=\tilde{d}_\tau(\tilde{A},\tilde{B})+\tilde{d}_\varepsilon(\tilde{A},\tilde{B}),$$

$$\tilde{d}\tau(\tilde{A},\tilde{B})=\frac{1}{2}\int_0^1\left\{(\underline{a}_\alpha-\underline{b}_\alpha)^+ +(\overline{a}_\alpha-\overline{b}_\alpha)^+\right\}d\alpha,\ (x)^+=\begin{cases} x & \text{if } x \ge 0 \\ 0 & \text{if } x<0 \end{cases},$$

$$\tilde{d}\varepsilon(\tilde{A},\tilde{B})=\frac{1}{2}\int_0^1\left\{(\underline{a}_\alpha-\underline{b}_\alpha)^- +(\overline{a}_\alpha-\overline{b}_\alpha)^-\right\}d\alpha,\ (x)^-=\begin{cases} 0 & \text{if } x \ge 0 \\ x & \text{if } x<0 \end{cases},$$

and $[\underline{a}_\alpha,\overline{a}_\alpha]$, $[\underline{b}_\alpha,\overline{b}_\alpha]$ are respectively, the $\alpha-\text{cut}$ of $\tilde{A}$ and $\tilde{B}$.

Second, we suppose that there are n independent jobs to be processed on a single machine. Each job j, j = 1,...,n requires fuzzy processing time p_j fuzzy due date d_j, which are a triangular fuzzy number (TFN). The machine can process at most job at a time, and the problem is to determine a sequence λ to process the jobs so that $\tilde{L}_{\max}$. The membership function $\tilde{P}_j(x)$ and $\tilde{D}_j(x)$ are defined in terms of three numbers $[p_j^1,p_j^c,p_j^u]$ and $[d_j^1,d_j^c,d_j^u]$ as follows:

$$\tilde{P}_j(x)=\begin{cases} 0 & \text{if } x<p_j^1 \\ \dfrac{(x-p_j^1)}{(p_j^c-p_j^1)} & \text{if } p_j^1 \le x<p_j^c \\ \dfrac{(p_j^u-x)}{(p_j^u-p_j^c)} & \text{if } p_j^c \le x<p_j^u \\ 0 & \text{if } p_j^u \le x \end{cases},$$

and

$$\tilde{D}_j(x)=\begin{cases} 0 & \text{if } x<d_j^1 \\ \dfrac{(x-d_j^1)}{(d_j^c-d_j^1)} & \text{if } d_j^1 \le x<d_j^c \\ \dfrac{(d_j^u-x)}{(d_j^u-d_j^c)} & \text{if } d_j^c \le x<d_j^u \\ 0 & \text{if } d_j^u \le x \end{cases}.$$

The possible range of the fuzzy processing time $\tilde{P}_j$ and fuzzy due date $\tilde{D}_j$ are $[p_j^1, p_j^u]$ and $[d_j^1, d_j^u]$, where the maximum value occurs at the point p_j^c and d_j^c respectively. Accordingly, the range $[p_j^1, p_j^u]$ and $[d_j^1, d_j^u]$ and the points p_j^c and d_j^c are called the supports and the cores of $\tilde{P}_j$ and $\tilde{D}_j$, respectively.

Assuming the processing time P_j and due date D_j are a crispy numbers for the time being then the cost function we are interested to study has the following form $\sum_{j=1}^{n} C_j + L_{\max}$. If the processing time and due date are a fuzzy numbers, then $\sum_{j=1}^{n}\tilde{C}_j+\tilde{L}_{\max}$ is a fuzzy function we denote the problem formulated in this sections $1/\tilde{P}_j = TFN,\ \tilde{D}_j = TFN/\sum\tilde{C}_j+\tilde{L}_{\max}(\tilde{C}_j,\tilde{D}_j)$. For find the formula of $\tilde{L}(\tilde{C}_j,\tilde{D}_j)$, we replacing fuzzy distance $\tilde{d}$ by $\tilde{L}$, $\tilde{A}$ with $\tilde{C}_j$, the fuzzy completion time of job j and $\tilde{B}$ with $\tilde{D}_j$ fuzzy due-date of the job j, the lateness of the job can be evaluated using the following lateness function:

$$\tilde{L}(\tilde{C}_j,\tilde{D}_j)=\tilde{L}_\tau(\tilde{C}_j,\tilde{D}_j)+\tilde{L}_\varepsilon(\tilde{C}_j,\tilde{D}_j)=$$

$$=\frac{1}{2}\int_0^1\left\{(\underline{c}_{j\alpha}-\underline{d}_{j\alpha})^+ + (\overline{c}_{j\alpha}-\overline{d}_{j\alpha})^+\right\}d\alpha+$$

$$+\frac{1}{2}\int_0^1\left\{(\underline{c}_{j\alpha}-\underline{d}_{j\alpha})^- + (\overline{c}_{j\alpha}-\overline{d}_{j\alpha})^-\right\}d\alpha,$$

where $[\underline{c}_{j\alpha},\overline{c}_{j\alpha}]$ and $[\underline{d}_{j\alpha},\overline{d}_{j\alpha}]$ are the $\alpha-$cut of $\tilde{C}_j$ and $\tilde{D}_j$ respectively.

Hence:

$$\tilde{L}(\tilde{C}_j,\tilde{D}_j)=\frac{1}{2}\int_0^1\Big\{\Big((c_j^c-c_j^1)\alpha+c_j^1-\big((d_j^c-d_j^1)\alpha+d_j^1\big)\Big)^+ +$$

$$+\Big(\big(-(c_j^u-c_j^c)\alpha+c_j^u\big)-\big(-(d_j^u-d_j^c)\alpha+d_j^u\big)\Big)^+\Big\}d\alpha+$$

$$+\frac{1}{2}\int_0^1\Big\{\Big((c_j^c-c_j^1)\alpha+c_j^1-\big((d_j^c-d_j^1)\alpha+d_j^1\big)\Big)^- +$$

$$\Big(\big(-(c_j^u-c_j^c)\alpha+c_j^u\big)-\big(-(d_j^u-d_j^c)\alpha+d_j^u\big)\Big)^-\Big\}d\alpha.$$

Then, the fuzzy lateness cost function for job j takes the following form:

$$\tilde{L}(\tilde{C}_j,\tilde{D}_j)=\frac{1}{4}\left(2c_j^c+c_j^l-2d_j^c-d_j^l+c_j^u-d_j^u\right).$$

The following are the basic characteristics of this function.

The (FEFDD) rule (fuzzy earliest fuzzy due-date rule) min $\tilde{L}_{\max}$. In this rule the jobs are ordered in non-decreasing order of their due-dates
$\left(d_{[1]}^l\le d_{[2]}^l\le\ldots\le d_{[n]}^l, d_{[1]}^c\le d_{[2]}^c\le\ldots\le d_{[n]}^c \text{ and } d_{[1]}^u\le d_{[2]}^u\le\ldots\le d_{[n]}^u\right)$.

3.1. Theorem

The maximum fuzzy lateness $\tilde{L}_{\max}$ is minimum by sequencing by (FEFDD) rule.

Proof: By using pairwise adjacent interchange jobs rule.

Consider the seq. $s=(\sigma ij\sigma')$ where σ and σ' are two partial sequence and i, j are two adjacent jobs with

$d_i^l>d_j^l,\ d_i^c>d_j^c$ and $d_i^u>d_j^u$, let

$p^l=\sum_{i\in\sigma}p_i^l,\ p^c=\sum_{i\in\sigma}p_i^c$ and $p^u=\sum_{i\in\sigma}p_i^u$,

and let $\tilde{L}$ be the max. lateness of the (n-2) jobs of σ and σ'.

Hence:

$$\tilde{L}_{\max}(s)=\max\{\tilde{L},\tilde{L}_i,\tilde{L}_j\},$$

where:

$$\tilde{L}_i=\frac{1}{4}\left(2(p^c+p_i^c-d_i^c)+p^l+p_i^l-d_i^l+p^u+p_i^u-d_i^u\right),$$

$$\tilde{L}_j=\frac{1}{4}\left(2(p^c+p_i^c+p_j^c-d_j^c)+p^l+p_i^l+p_j^l-d_j^l+p^u+p_i^u+p_j^u-d_j^u\right).$$

Now consider the new sequence $s'=(\sigma ij\sigma')$, the fuzzy completion times of all the jobs of σ and σ' are the same,

$$\tilde{L}'_{\max}(s')=\max\{\tilde{L},\tilde{L}'_i,\tilde{L}'_j\},$$

where

$$\tilde{L}'_j=\frac{1}{4}\left(2(p^c+p_j^c-d_j^c)+p^l+p_j^l-d_j^l+p^u+p_j^u-d_j^u\right),$$

$$\tilde{L}'_i=\frac{1}{4}\left(2(p^c+p_j^c+p_i^c-d_i^c)+p^l+p_j^l+p_i^l-d_i^l+p^u+p_j^u+p_i^u-d_i^u\right),$$

and

$$\tilde{L}_j>\tilde{L}'_j \text{ and } \tilde{L}_j>\tilde{L}'_i \text{ since } d_i^l>d_j^l,\ d_i^c>d_j^c \text{ and } d_i^u>d_j^u.$$

Hence

$$\tilde{L}'_j>\max\{L'_i,L'_j\},$$

Then

$$L'_{\max}(s)\ge L'_{\max}(s').$$

By repeating this procedure, we get that the FEFDD rule is min. $L'_{\max}$.

3.2. Theorem

All jobs can be completed on time if and only if the FEFDD rule gives $\tilde{L}_{\max}=0$.

Proof: All the jobs are completed on time (n jobs)

iff $c_j^l\le d_j^l,\ c_j^c\le d_j^c$ and $c_j^u\le d_j^u\ \forall$ j=1,2,...,n,

iff $c_j^l-d_j^l\le 0,\ c_j^c-d_j^c\le 0$ and $c_j^u-d_j^u\le 0,\ \forall$ j=1,2,...n,

But $L_j=\frac{1}{4}\left(2(c_j^c-d_j^c)+c_j^l-d_j^l+c_j^u-d_j^u\right)$,

$\tilde{L}_{\max}\le 0,\ \forall$ j=1,2,...,n,

iff there exist at least one job j which

$c_j^l=d_j^l,\ c_j^c=d_j^c$ and $c_j^u=d_j^u$ i.e.

iff $\tilde{L}_{\max}=0$.

4. Methodology

4.1. Local Search Techniques

In this section we study local search techniques which are useful tools for solving single machine scheduling problem $1/\tilde{P}_j = \text{TFN},\ \tilde{D}_j = \text{TFN}/\sum \tilde{C}_j + \tilde{L}_{\max}\left(\tilde{C}_j, \tilde{D}_j\right)$.

Local search is an iterative algorithm that moves from one solution s to another s' according to some neighborhood structure.

Local search procedure usually consists of the following steps.

1. Initialization. Choose an initial schedule s to be the current solution and compute the value of the objective function F(s).

2. Neighbor Generation. Select a neighbor s' of the current solution s and compute $F(s')$.

3. Acceptance Test. Test whether to accept the move from s to s'. If the move is accepted, then s' replaces s as the current solution; otherwise s is retained as the current solution.

4. Termination Test. Test whether the algorithm should terminate. If it terminates, output the best solution generated; otherwise, return to the neighbor generation step.

We assume that a schedule is represented as a permutation of job numbers $(j_1, j_2, \ldots, j_n)$. This can always be done for a single machine processing system or for permutation flow shop; for other models more complicate structures are used.

In step (1), a starting solution can be obtained by one of the constructive heuristics described in the previous lectures or it can be specified by a random job permutation. If local search procedure is applied several times, then it is reasonable to use random initial schedules.

To generate a neighbor s' in step (2), a neighborhood structure should be specified beforehand. Often the following types of neighborhoods are considered:

• *transpose neighborhood* in which two jobs occupying adjacent positions in the sequence are interchanged:

(1, **2**, **3**, 4, 5, 6, 7) → (1, **3**, **2**, 4, 5, 6, 7);

• *swap neighborhood* in which two arbitrary jobs are interchanged:

(1, **2**, 3, 4, 5, **6**, 7) → (1, 6, 3, 4, 5, 2, 7);

• *insert neighborhood* in which one job is removed from its current position and inserted elsewhere:

(1, **2**, 3, 4, 5, 6, 7) → (1, 3, 4, 5, 6, **2**, 7).

Neighbors can be generated randomly, systematically, or by some combination of the two approaches.

In step (3), the acceptance rule is usually based on values F(s) and $F(s')$ of the objective function for schedules s and s'. In some algorithms only moves to 'better' schedules are accepted (schedule s' is better than s if $F(s') < F(s)$); in others it may be allowed to move to 'worse' schedules. Sometimes "wait and see" approach is adopted.

The algorithm terminates in step (4) if the computation time exceeds the prespecified limit or after completing the prespecified number of iterations.

4.2. Threshold Acceptance Method (TH)

A variant of simulated annealing is the **threshold acceptance method** (Brucker 2007). It differs from simulated annealing only by the acceptance rule for the randomly generated solution $s' \in N$. s' is accepted if the difference $F(s') - F(s)$ is smaller than some non-negative threshold t. t is a positive control parameter which is gradually reduced. Fig. 1 shows the generic implementation of Threshold acceptance structure.

```
While (termination condition in not satisfied) do
New solution ← neighbors (best solution);
If new solution is better than actual solution then
Best solution ← actual solution
Elseif difference between old and new solution less than control
parameter t then
Best solution ← actual solution
End if

End while
```

Fig. 1. Threshold acceptance structure

The threshold acceptance method has the advantage that they can leave a local minimum. They have the disadvantage that it is possible to get back to solutions already visited. Therefore oscillation around local minima is possible and this may lead to a situation where much computational time is spent on a small part of the solution set.

4.3. Tabu Search (TS)

The use of the tabu search was pioneered by Glover [8] who from 1994 onwards has published many articles discussing its numerous applications. Others were quick to adopt the technique which has been used for such purposes as sequencing, scheduling, oil exploration and routing.

The properties of the tabu search can be used to enhance other procedure by preventing them becoming stuck in the regions of local minima. The tabu search utilizes memory to prevent the search from returning to a previously explored region of the solution space too quickly. This is achieved by retaining a list of possible solutions that have been previously encountered. These solutions are considered tabu-hence the name of the technique. The size of the tabu list is one of the parameters of the tabu search. The tabu search also contains mechanism for controlling the search. The tabu list ensures that some solution will be unacceptable; however, the restriction provided by the tabu list may become too limiting in some cases causing the algorithm to become trapped at a locally optimum solution. The tabu search introduces the notion of aspiration criteria in order to overcome this problem. The aspiration criteria over-ride the tabu restrictions making it possible to broaden the search for the global optimum.

An initial solution is generated (usually randomly). The tabu list is initialized with the initial solution. A number of iterations are performed which attempt to update the current solution with a better one, subject to the restriction of the tabu list. A list of candidate solution is proposed in every iteration. The most admissible solution is selected from the candidate list. The current solution is updated with the most admissible one and the new current solutions added to the tabu list. The algorithm stops after a fixed number of iterations or when a better solution has been found for a number of iterations. Fig. 2 shows the generic implementation of tabu search.

```
S = Generate Initial Solution ()
Initialize Tabu List ($TL_1, \ldots, TL_r$)
K = 0
While (termination condition in not satisfied) do
    Allowed Set (S, K) = {$z \in N(s)$ no tabu condition is violated or at least one
    Aspiration criterion is satisfied}
    S = Best Improvement (S, Allowed Set(S, K))
    Update Tabu List and Aspiration Condition ()
    K = K+1
End while
```

Fig. 2. A generic Tabu Search

4.4. Memetic Algorithm Approach (MA)

Memetic algorithms (MA), combines the recognized strength of the population-based methods with the intensification capability of a local search. In an MA, all individuals of the population evolve solutions until they become local minima of a certain neighborhood (or highly evolved solutions of individual search strategies), i.e., after the recombination and mutation steps, a local search is applied to the resulting solutions. A more formal introduction to MA and polynomial merger algorithms can be found in Moscato [15]. Fig. 3 shows a pseudo-code representation of a local search-based memetic algorithm.

```
Procedure Local Search-Based Memetic Algorithm;
    BEGIN
    Initialize Population Pop using First Pop ();
    For Each individual i ∈ Pop DO  i:=Local-Search (i);
    For Each individual i ∈ main Pop DO Evaluate Fitness (i);
        REPEAT /*generation loop */
        FOR i:= 1 to #recombination's DO
            Select to merge a set S_par ⊆ Pop;
            Offspring: = Recombine (S_par, x);
            IF (select To Mutate offspring) THEN offspring := Mutate
(offspring);
            offspring:= Local-Search (offspring);
            Evaluate Fitness (offspring);
            Add in Population individual offspring to Pop;
        End For;
        IF (Pop has_converged) Pop:= RestartPop(Pop);
    UNTIL stop criterion;
END
```

Fig. 3. Pseudo-code of a Memetic Algorithm

The initialization part begins at **initialize Population** and ends just before the **repeat** command. This part is responsible for the generation, optimization and evaluation of the initial population (*Pop*). The second part includes the so-called 'generation loop'. At each step, two parent configurations are selected for recombination and an offspring is produced and, if selected to mutate, it suffers a mutation process. The next steps are local search, evaluation and insertion of the new solution into the population. If the population is considered to have lost diversity, a mutation process is applied on all individuals except the best one. Finally, a termination condition is checked.

4.4.1. Population Structure

In our implementation we use a hierarchically structured population organized as a complete ternary tree of individuals clustered in 4 subpopulations or clusters, as shown in figure(4). In contrast with a non-structured population it restricts crossover possibilities. Other studies have shown that the use of structured populations is more effective when compared to non-structured populations (e.g. França *et al.*(5)).

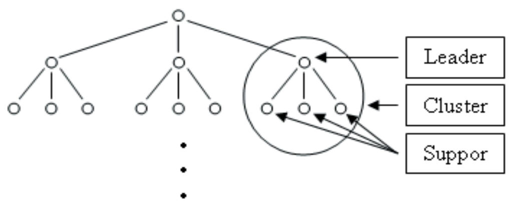

Fig. 4. Population structure

The structure consists of several clusters, each one composed of a leader and three supporter solutions. The leader of a cluster is always better fitted than its supporters. This hierarchy ensures top clusters have better fitted individuals than bottom clusters. As new individuals are constantly generated, replacing old ones, periodic adjustments to keep this structure well-ordered are necessary.

The number of individuals in the population is restricted to the numbers of nodes in a complete ternary tree: 13, 40, 121, etc. That is, 13 individuals are necessary to construct a ternary tree with 3 levels, 40 to one with 4 levels and so on.

4.4.2. Representation of Individuals

The representation we have chosen for the $1/\tilde{P}_j = \mathrm{TFN},\ \tilde{D}_j = \mathrm{TFN}/\sum \tilde{C}_j + \tilde{L}_{\max}\left(\tilde{C}_j, \tilde{D}_j\right)$ is quite intuitive, with a solution represented as a chromosome with the alleles assuming different integer values in the [1, n] interval, where n is the number of jobs. There are m-1cut-points in the chromosome that define the subsequences assigned on machine.

For instance, < 4 9 6 * 2 8 5 1 * 3 10 7 > is a possible solution for a problem with 10 jobs. The cut-points (*) are in positions 4 and 9. Therefore, subsequence 1 executes operations 4 - 9 - 6, in this order; subsequence 2 executes operations 2 - 8 - 5 - 1 and subsequence 3 performs operations 3 - 10 - 7.

4.4.3. Recombination

The command **select To Merge** indicates the task of selecting a subset of individuals (called $S_{par} \subseteq \mathrm{Pop}$) to be used as input for the crossover operation, represented by the Recombine( ) function. In the pseudo code, the symbol 'x' stands for the instance of the problem. In this case, since we are addressing the $r_j \left| \sum_j w_j C_j \right.$, the 'x' refers to matrix s_{ij} and vector p_j.

The crossover operator implemented is the well-known Order Crossover (OX). After choosing two parents, a fragment of the chromosome from one of them is randomly selected and copied into the offspring. In the second phase, the offspring's empty positions are sequentially filled according to the chromosome of the other parent.

Parent	A 2 4 * 7 6 3 * 1 5
Parent B	6 5 2 * 7 1 4 * 3
Initial Offspring	1 1 1 7 6 3 * 1 1 (A)
Construction phase	5 2 2 7 6 3 * 2 1 (B)
	5 2 2 7 6 3 * 2 1 (B)
	5 2 * 7 6 3 * 2 1 (B)
	5 2 * 7 6 3 * 1 1 (B)
Final Offspring	5 2 * 7 6 3 * 1 4 (B)

In the example above, the fragment is selected from the parent A and consists of the alleles < 7 6 3 * >. The child's empty positions were then filled according to the order that the alleles appear in the chromosome of parent B. The number of new individuals generated in every iteration is controlled by a parameter named *cross rate* which is expressed as the percentage of new individuals over the total population.

4.4.4. Mutation

In our method, a traditional mutation strategy based on job swapping was implemented. According to it, two positions are randomly selected and the alleles in these positions swap their values.

The alleles that are swapped can be both related to two jobs (two integers) or one to a job and other to a cut-point. In the first case the number of jobs on each machine remains the same. In the second case the structure of the solution is changed, because the number of jobs on each machine is modified. The case in which both positions selected are cut-points does not change anything at all.

We implemented two mutation procedures - Mutate ( ) and Restart Pop( ); the first can be considered a light mutation and the other is a heavy mutation procedure. The Mutate( ) function is applied to each individual with a probability of *mut_rate* and, once applied, it mutates two alleles. Implementations with more changes per individual showed no improvement.

In fact, when the number of alleles to be mutated increases, valuable information tends to be lost, worsening the MA's overall performance. The Restart Pop( ) procedure, on the other hand, mutates all individuals in the *main Pop* except the incumbent solution. The swapping procedure is applied to each individual 10n times, so the resulting population almost resembles a randomized restarting procedure.

4.4.5. Fitness Function

As in this problem the goal is to minimize the maximum fuzzy lateness and the maximum completion time, the fitness function was chosen as randomly.

4.4.6. Selection of Parents

Recombination is only allowed between a leader and one of its supporters and both are randomly selected. An intensification procedure was implemented, forcing the best individual to take part in approximately 10% of the crossovers.

This procedure showed itself to be very effective when compared to a standard selection policy. Tests revealed small but repeated improvements over the scheme without intensification.

4.4.7. Offspring Insertion into Population

Once the leader and one supporter are selected, the recombination, mutation and local search take place and an offspring is generated.

If the fitness of the offspring is better than the supporter's that took part in the recombination, the offspring replaces the supporter. If the new individual is already present in the population, it is not inserted in it. We adopted a policy of not allowing duplicated individuals to reduce loss of diversity.

After the generation is over and all individuals were inserted, the population is restructured.

The hierarchy forces the fitness of an individual to be lower than the fitness of the individual just above it in the ternary tree. Following this policy, the higher clusters will have leaders with better fitness than the lower clusters and the best solution will be the leader of the root cluster.

The adjustment is made by comparing each individual to the individual just above which it is connected to. If the lower individual becomes better than the upper one, they swap places.

5. Computational Results

Local search methods were tested by coding then in Matlab R2009b and runs on a Pentium IV at 2.20GHz, 2.0GB computer. The tested problem instances are generated as follows:

For n=10, 20, 30, 50, 100, 200, 500 & 1000. The fuzzy processing times were generated uniformly with support in the range of [10, 30] and the fuzzy due dates were generated randomly with the support in the range of [1, W], where W was also randomly selected among the values of {10, 20, 30, 40, 50}.

The efficiency local search heuristic methods (Threshold accepted (TA), Tabu search (TS), and Memetic algorithm (MA)) have been approached in terms of comparable rate of value. In the following Tab. 1a for MA we compared among three ways for selection the best values for fitness function first by choose the minimum values for c and we denoted MAc, and second choose by use the ratio $r_6 = {(1+4c+u)}/{6}$ (like the Program Evaluation and Review Technique (PERT) in network models) where l,u are called the lower and upper limits of support and c is called the core of triangular fuzzy number we denoted MA r_6. Finally we suggested another comparison between fuzzy numbers by using a ranking function $r_6 = {(1+4c+u)}/{6}$ and took symbol MA r_4. Tab. 1a show that the MA r_4 gives best solution for l and u.

Tab. 1b was used the same ways for selection (TSc, TS r_4 and TS r_6). Tab. 1b show that for jobs (10, 20, 30) was instable although TS r_6 was some time better. TS r_4 gives best solution for l and u for equal or greater than 50 jobs.

Table 1a

Compares values for MAc, MA r_4 and MA r_6

		10	20	30	50	100	200	500	1000
MAc	l	890.75	5270	16311.65	70651.35	710601.93	4899545.9	68366984.23	730663074.4
	c	1086.65	6152.7	18235.55	75890.75	911930.13	5730969.7	73912322.33	962023768
	u	1449.95	7617.7	21154.85	83503.65	1214757.33	6898728	80938418.93	1234143921
MA r_4	l	888.85	5249.48	16182.2	70231.23	696142.2	4856359.08	68169184.8	727746820.53
	c	1092.75	6170.98	18249.2	75944.43	915795.8	5748871.18	73982833.6	964482995.73
	u	1434.55	7536.98	21009.3	83079.73	1203876.6	6847733.08	80732460.7	1228511021.53
MA r_6	l	888.375	5264.05	16220.68	70408.3	699666.1	4864920.45	68257407.15	727893492.2
	c	1087.18	6165.85	18231.48	76005.8	912194.9	5730921.25	73948652.15	962318409.1
	u	1436.98	7567.75	21042.98	83182.9	1208130.4	6873285.35	80815578.85	1229680502

Table 1b

Compares values for TSc, TS r_4 and TS r_6

		10	20	30	50	100	200	500	1000
TSc	L	881.93	5255.2	16231.05	70376.43	702286.25	4916151.05	68603791.15	732545060.58
	C	1128.33	6319.4	18614.05	77137.73	956784.15	5923156.65	74937674.05	983947263.78
	u	1433.33	7519.3	20980.65	83230.13	1208304.85	6905950.85	81223058.05	1235212742.18
TS r_4	l	885.98	5253.45	16217.18	70243.38	700640.53	4864997.15	68427077.98	729213883.3
	c	1089.58	6186.05	18294.98	76115.88	919805.23	5775052.95	74441493.78	969323757.9
	u	1433.18	7534.55	20992.68	83069.38	1206265.43	6880457.35	81044918.78	1231395664
TS r_6	l	889.13	5249.03	16199	70352.23	705715.33	4884784.48	68450579.3	730378090.9
	c	1088.33	6159.93	18247.7	76156.23	916788.93	5768822.78	74362071	967491138.6
	u	1443.23	7563.83	21036.2	83230.23	1210502.63	6887176.58	81078921.7	1233783933.5

Tab. 1c show that the TH r_4 gives best solution for l and u. And TH r_6 gives best solution for c.

Table 1c

Compares values for THc, TH r_4 and TH r_6

		10	20	30	50	100	200	500	1000
THc	l	892.68	5320.4	16433.4	70635.75	721921.53	4930327.55	68483229.03	732908423.03
	c	1109.88	6328.3	18722.7	77184.85	962902.93	5907957.15	74631257.53	979749112.73
	u	1428.58	7572.4	21187.2	83565.65	1225292.23	6928371.85	81079157.33	1235686296.23
TH r_4	l	886.18	5242.53	16201.18	70307.73	699981.35	4872741.85	68449639.25	728205961.65
	c	1089.98	6185.73	18265.28	76117.23	922723.65	5779776.25	74449373.15	969318624.95
	u	1432.18	7532.93	21016.48	83095.73	1204659.15	6882313.65	81053998.05	1231711787.85
TH r_6	l	886.23	5252.55	16225.58	70459.18	703683.93	4878116.6	68466508.7	730178380.28
	c	1089.53	6167.95	18272.98	75939.88	916443.63	5764499.5	74379550.9	967304040.58
	u	1439.93	7559.65	21045.98	83260.88	1211811.23	6887759.8	81080858.6	1232740610.48

In the following Tab. 2 we took the best results from the Tab. 1a, 1b, 1c and show the efficiency local search heuristic methods (MA r_4, TS r_4 and TH r_4) have been approached in terms of comparable rate of values. MA r_4 gives the best values for the big jobs but for the small jobs the results was unclear what is the approach gives best solution.

Table 2

Compares values of local search methods

		10	20	30	50	100	200	500	1000
MA r_4	l	888.85	5249.475	16182.2	70231.225	696142.2	4856359.075	68169184.8	727746820.53
	c	1092.75	6170.975	18249.2	75944.425	915795.8	5748871.175	73982833.6	964482995.73
	u	1434.55	7536.975	21009.3	83079.725	1203876.6	6847733.075	80732460.7	1228511021.53
TS r_4	l	885.98	5253.45	16217.18	70243.38	700640.53	4864997.15	68427077.98	729213883.3
	c	1089.58	6186.05	18294.98	76115.88	919805.23	5775052.95	74441493.78	969323757.9
	u	1433.18	7534.55	20992.68	83069.38	1206265.43	6880457.35	81044918.78	1231395664
TH r_4	l	886.18	5242.53	16201.18	70307.73	699981.35	4872741.85	68449639.25	728205961.65
	c	1089.98	6185.73	18265.28	76117.23	922723.65	5779776.25	74449373.15	969318624.95
	U	1432.18	7532.93	21016.48	83095.73	1204659.15	6882313.65	81053998.05	1231711787.85

In the following Tab. 3 show the efficiency local search heuristic methods (MA r_4 , TS r_4 and TH r_4) have been approached in terms of comparable rate of times. TH r_4 gives the best times for the all jobs then TS r_4 was butter than MA r_4 .

Table 3

Compares times of local search methods

	10	20	30	50	100	200	500	1000
M r_4	0.213938	0.321157	0.433444	0.651585	1.232433	2.532499	7.644073	18.78292
TS r_4	0.023767	0.030195	0.02531	0.026875	0.03158	0.042565	0.087396	0.326691
TH r_4	0.023512	0.02302	0.024426	0.025027	0.027128	0.030516	0.041107	0.060182

6. Concluding Remarks

We have developed a new model to formulate the situation where jobs with fuzzy processing times and fuzzy due dates are to be scheduled on a single machine. The local search methods used to solve all the large problems the result show the robustness and flexibility of local search heuristics.

References

1. Brucker, P. (2007). “Scheduling Algorithms”, Springer Berlin Heidelberg New York, Fifth Edition.
2. Chanas, S., Kasperski, A. (2001). “Minimizing maximum lateness in a single machine scheduling problem with fuzzy processing times and fuzzy due dates”, Engineering Application of Artificial Intelligence, 14, 377-386.
3. Chanas, S., Kasperski, A. (2003). “On two single machine scheduling problems with fuzzy processing times and fuzzy due dates”, European Journal of Operational Research, 147, 281-296.
4. Dubois, D., Prade H. (1980). “Fuzzy sets and systems: theory and applications”, Academic press, Inc.
5. França, P. M. (1999). Mendes A. and Moscato P., “A memetic algorithm for the total tardiness single machine scheduling problem” European Journal of OR.
6. Glover, F. (1994). “A user’s guide to tabu search”, Annals of Operations Research, 41, 3-28.
7. Goguen, J. (1973). “The Fuzzy Tychonoff Theorem”, J. Math. Anal. Appl, 43, 734-742.
8. Han, S., Ishii, H., Fujii, S. (1994). “One machine scheduling problem with fuzzy due dates”, European Journal Operation. Research , 79, 1-12.
9. Heilpern, S. (1997). “Representation and application of fuzzy numbers”, Fuzzy Sets and Systems, 91, 259-268.
10. Hussam, A., Faria, A., Hanan, A. and Fathalh, A. “On Fuzzy Distances and their Applications on Cost Function $\tilde{L}_{\max}\left(\tilde{C}_j,\tilde{D}_j\right)$”, to appear.
11. Ishii, H., Tada, M. (1995). “Single machine scheduling problem with fuzzy precedence relation”, European Journal of Operational Research, 87, 284-288.
12. Kaufmann, A., Gupta, M. (1991). “ Introduction to fuzzy arithmetic theory and applications”, Van Nostrand Reinhold.
13. Lam, S., Cai X. (1999). “Distance measures of fuzzy numbers computational intelligence and applications”, Springer, Berlin, 207-214.
14. Lam, S., Cai, X. (2002). “Single machine scheduling with nonlinear lateness cost functions and fuzzy due dates nonlinear analysis “, Real World Application, 3333, 307-31.
15. Moscato, P. (1989). “On evolution, search, optimization, genetic algorithm and martial arts: toward memetic algorithms”, Caltech Concurrent Computation Program c3p Technical Report, California, 826.
16. Nachammai, A. (2012). “Solving fuzzy linear Fractional programming problem using Metric distance Ranking”, Applied Mathematical Sciences, 6, 1275-1285.
17. Tanaka, K., Vlach, M. (1997). “Single machine scheduling with fuzzy due dates”, Seventh IFSA World Congress Prague, 195-199.
18. Zadeh, L. (1965). “Fuzzy sets” Information and Control, 8, 338-353.