Scheduling job families with setups on a single machine

Hussam Abid Ali Mohammed
Department of Mathematics, College of Education, University of Kerbala, Kerbala, Iraq,

Abstract

Many sequencing problems have a combinatorial nature and they are very difficult to optimality within acceptable computation times.
We consider the problem of scheduling jobs on a single machine minimize the maximum completion time and maximum weighted earliness time. The jobs partitional into families, and a setup time is necessary for scheduling the fest job and when there is a switch in processing jobs from one family to jobs of another family. And to solve this problem we suggested method heuristic to compare and test different local search method.

المستخلص

هنالك عدة مسائل في الجدولة تمتلك الصيغة التوافقية وهذه المسائل من الصعب جداً إيجاد الحل الأمثل لها خلال أوقات حسابية معقولة.

لقد تناولنا مسألة جدولة النتاجات على ماكنة واحدة لتصغير دالة الهدف وتكبير وقت الإكمال (maximum completion time and maximum weighted earliness time). لقد قسمت النتاجات الى f من العوائل وهناك وقت إعداد ضروري للماكنة عند جدولة أول نتاج وعند جدولة نتاج من عائلة تختلف عن عائلة النتاج الذي سبقه. ولحل هذه المسألة تم اقتراح طريقة تقريبية (heuristic method) للحصول على حل قريب من الحل الأمثل (near optimal solution).

1. Introduction

Consider a production to order system in which the product range can be decomposed to a number of product families. Between production of orders for products belonging to the same family almost no setup is required, where as a serious setup is incurred between orders for products belonging to different families. Hence for reasons of efficiency, we prefer to continue with orders for products belonging to the same family as long as possible.

However the need to finish order as class as possible to their required due date may conflict with the efficiency.

We consider a single machine production system in which production is to order. Jobs are characterized by their type (the family the belong to) their processing times. We assume that jobs of the same family have the same setup time. Each time we start producing jobs of a type different from the one just completed, a setup is required.

Many practical-scheduling problems involve sequencing number of jobs divided into several families, by a machine set-up times. Most of these problems are Np-hard, even without setup times and thus are difficult to solve to optimality. Various scheduling problems in manufacturing and service organizations can be formulated as single facility problems with job classes (families). For example Bruno and Downey [4], Monma and Potts [12] and Chen [6] described a computer system application, in which computer jobs requires different compilers. A setup is not incurred if the next job requires a compiler that is already resident in memory. However, if the next job requires a non-resident compiler, a setup time that depends only on the time needed to load the new compiler is incurred [3]. Little work has been done on scheduling problems multiple objectives with and without setup times. Abdul-Razaq and Potts [2], Moghaddam and et. al. [11], discussed a scheduling problem without setup times to minimize the total cost of earliness and tardiness. Van-Wassenhove and Gelders [15] discussed scheduling problem without setup times to minimize the sum of completion time and the maximum tardiness, where the objective is to minimize two different criteria.

In this paper we will use heuristic methods to solve the problem of scheduling number of jobs *n*, ($N = \{1, \dots, n\}$) on a single machine where the machine can process only one job at a time, and it is permanently available at time zero, no idle time is permitted, no preemption of jobs is allowed and all the jobs are initially available. The jobs are divided into several families a setup time for a job from family *f* if it is the first job in the schedule or if it is scheduled immediately after a job from different family, the machine cannot perform any processing while undergoing a setup.

Each family *f*, for $f = 1, \dots, F$, contains n_f jobs. A job *i* in the *f-th* family can be denoted as (*i, f*), each job (*i, f*) ($i = 1, \dots, n_f$, $f = 1, \dots, F$) has a processing time p_{if} the problem to find a feasible schedule with minimum maximum completion time and maximum weighted earliness time with setup time $C_{\max} + E^w_{\max}$ the problem, denoted as $1\left|S_f\right|C_{\max} + E^w_{\max}$.

2. Problem Formulation

The scheduling groups of jobs on a single machine problem can be described as follows: we are given *N* jobs that are divided into families. Each family *f*, for $1 \le f \le F$, contains n_f jobs. Sometimes it is more convenient to refer to job (*i, f*) which is the *i-th* job in *f*, for $1 \le i \le n_f$.

All jobs are available for processing at time zero, and they are to be scheduled on single machine. We let p_{if} denote the processing time of job (*i, f*). A machine setup time S_f is incurred whenever a job in family *f* is processed, immediately after a job in different family. Also, a setup time S_f is required for processing the first job in the scheduling.

Suppose the processing order $\sigma = (\sigma(1), \dots, \sigma(n))$ a vector $(\delta_{\sigma(1)}, \dots, \delta_{\sigma(n)})$ of corresponding setup times is easily constructed. The setup time required immediately before the processing of job $\sigma(i), (i = 1, \dots, n)$ is given by:

$\delta_{\sigma(1)}$: is the setup time of the first job (positive integer constraint).

$$\delta_{\sigma(i)} = \begin{cases} \alpha_{fg} & if \quad i > 1, \sigma(i-1) \in f \ \ and \ \ \sigma(i) \in g,\ f \ne g, g \in f \\ 0 & o.w. \end{cases}$$

Where α_{fg} is a positive integer constant.

Our object is to find a sequence with an associated completion time $C_{\sigma(i_f)}$ and weighted earliness time $w_{\sigma(i_f)} E_{\sigma(i_f)}$ for each job (*i, f*) that minimize the maximum completion time and maximum weighted earliness time $C_{\max} + E^w_{\max}$. This problem denoted by (P_f) can be stated as follows:

$$\left.\begin{array}{lll}
\min Z = C_{\max} + E^w_{\max} & & \\
\quad Subject\ to & & \\
p_{\sigma(i_f)} > 0 & i = 1, \dots, n_f, & f = 1, \dots, F \\
C_{\sigma(i_f)} = p_{\sigma(i_f)} & i = 1, & f = 1, \dots, F \\
C_{\sigma(i_f)} = C_{\sigma((i-1)_f)} + p_{\sigma(i_f)} & i > 1, i = i-1, & f = 1, \dots, F \\
C_{\sigma(i_f)} = C_{\sigma((i-1)_f)} + p_{\sigma(i_f)} + S_f & i > 1, i \ne i-1, & f = 1, \dots, F \\
C_{\max} = C_{\sigma(i_f)} & i = n, & f = 1, \dots, F \\
E_{\sigma(i_f)} = \max\limits_{1 \le i \le n_f} \left\{ d_{\sigma(i_f)} - C_{\sigma(i_f)}, 0 \right\} & i = 1, \dots, n_f, & f = 1, \dots, F \\
E_{\sigma(i_f)} \ge d_{\sigma(i_f)} - C_{\sigma(i_f)} & i = 1, \dots, n_f, & f = 1, \dots, F \\
E_{\sigma(i_f)} \ge 0 & i = 1, \dots, n_f, & f = 1, \dots, F \\
E^w_{\max} = \max\limits_{1 \le i \le n_f} \left\{ w_{\sigma(i_f)} E_{\sigma(i_f)} \right\} & i = 1, \dots, n_f, & f = 1, \dots, F
\end{array}\right\} (P_f)$$

Where σ is a schedule, $\sigma = (\sigma(1), \dots, \sigma(n_f)), \sigma \in \delta$ and δ is the set of all schedules and $d_{\sigma(i_f)}$ and $w_{\sigma(i_f)}$ denoted the due date and weighted jobs respectively and S_f is the setup time.

3. Initial Solution and Special Cases

3.1 Initial Solution (Ini)

When we start with the initial solution before using the local search means we start with good solution may be gives the optimal solution**.**

Lemma (1): There is permutation on machine *M*, given by non-increasing order of their weighted slack time $s_j = (d_j - p_j)/w_j$ which minimizing $E^w_{\max}$ (MSWT).

Lemma (2): For heuristic MSWT [the sequencing the job in non-increasing order of their weighted slack time $s_j = (d_j - p_j)/w_j$] each of the first $\lceil n/f \rceil$ jobs scheduled has earliness which don't exceed $E^w_{\max}$.

3.2 Special Cases

Finding a special case for scheduling problem means finding an optimal schedule directly without using BAB method or DP algorithm. A special case depends on satisfying some conditions in order to make the problem easily solved.

Case (1): For the problem P_f, if the all jobs are late then we ordered the sequence σ by non-increasing setup times give optimal solution.

Proof: Suppose we have non-increasing setup times sequence $\sigma = (1, 2, \ldots, n)$

$E_{\sigma(i_f)} = 0$, since $d_{\sigma(i_f)} \le C_{\sigma(i_f)}$ for all $i = 1, \ldots, n_f$ and $f = 1, \ldots, F$.

$$\min Z = C_{\max} + E^w_{\max} = C_{\max} + 0 = C_{\max}$$

That implies the non-increasing setup times sequence gives optimal solution for the problem $1|S_f|C_{\max}$ ■

Case (2): For the problem P_f, if we have the sequence σ satisfy the weighed slack time and non-increasing setup times then σ give optimal solution.

Proof: By lemma (1) and case (1) ■

4. Memetic Algorithm Approach

Memetic algorithms (MAs) (Moscato, 1989), 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 MAs and polynomial merger algorithms can be found in Moscato (1999). Figure 1 shows a pseudo-code representation of a local search-based memetic algorithm.

```
1.  procedure Local Search-based Memetic Algorithm;
      BEGIN
2.        Initialize Population Pop using First Pop();
3.        For Each individual $i \in Pop$ DO $i :=$ Local-Search($i$);
4.        For Each individual $i \in$ main Pop DO Evaluate Fitness($i$);
        REPEAT /*generation loop */
5.          FOR $i :=$ 1 to #recombinations DO
6.              Select To Merge a set $S_{par} \subseteq Pop$;
7.              offspring := Recombine($S_{par}$, $x$);
8.              IF (select To Mutate offspring) THEN offspring := Mutate (offspring);
9.              offspring := Local-Search(offspring);
10.             Evaluate Fitness(offspring);
11.             Add In Population individual offspring to Pop;
12.         End For;
13.         IF (Pop has_converged) Pop := RestartPop(Pop);
        UNTIL stop criterion;
    END
```

Figure 1. 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.

4.1 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 2. 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.* 1999; Buriol *et al.* 1999).

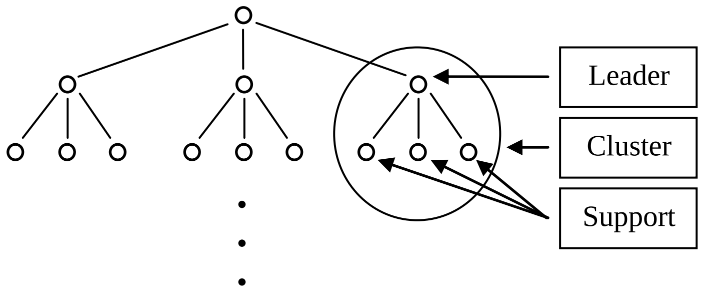

Figure 2. Population structure

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.

4.2 Representation of Individuals

The representation we have chosen for the $1|S_f|C_{\max}+E_{\max}^{w}$ 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*-1 cut-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.

4.3 Recombination

The command **selectToMerge** indicates the task of selecting a subset of individuals (called $S_{par} \subseteq Pop$) to be used as input for the crossover operation, represented by the Recombine( ) function. In the pseudocode, the symbol '*x*' stands for the instance of the problem. In this case, since we are addressing the $1|S_f|C_{\max}+E_{\max}^{w}$, 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	_ _ _ 7 6 3 * _ _ (A)
Construction phase	5 _ _ 7 6 3 * _ _ (B)
	5 2 _ 7 6 3 * _ _ (B)
	5 2 * 7 6 3 * _ _ (B)
	5 2 * 7 6 3 * 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 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 RestartPop( ); 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 RestartPop( ) procedure, on the other hand, mutates all individuals in the *mainPop* except the incumbent solution. The swapping procedure is applied to each individual 10n times, so the resulting population almost resembles a randomized restarting procedure.

4.5 Fitness Function

As in this problem the goal is to minimize the maximum completion time and maximum weighted earliness time with setup times, the fitness function was chosen as randomly.

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.

4.7 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. 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 $Z(s') - Z(s)$ is smaller than some non-negative threshold *t*. *t* is a positive control parameter which is gradually reduced.

Algorithm Threshold Acceptance

```
1.  $i := 0$;
2.  Choose an initial solution $s \in S$;
3.  $best := Z(s)$;
4.  $s^* := s$;
    REPEAT /*generation loop */
5.      Generate randomly a solution $s' \in N(s)$;
6.      IF $Z(s') - Z(s) < t_i$ THEN $s := s'$;
7.          IF $Z(s')$ < $best$ THEN
        BEGIN
8.          $s^* := s'$;
9.          $best := Z(s')$;
        END;
10.     $t_i + 1 := g(t_i)$;
11.     $i := i + 1$
      UNTIL stop criterion;
    END
```

g is a non-negative function with $g(t) < t$ for all *t*.

Figure 3. 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.

6. Tabu Search (TS)

In this section we describe the tabu search procedure used to solve the P_f. Tabu search (see Glover and Laguna [1997] and Gendreau [2003]) is one of the most popular techniques to find near optimal solutions to hard combinatorial optimization problems. A simple way to avoid such problems is to store all visited solutions in a list called tabu list T and to only accept solutions which are not contained in the list. However, storing all visited solutions in a tabu list and testing if a candidate solution belongs to the list is generally too consuming, both in terms of memory and computational time.

To make the approach practical, we store attributes which define a set of solutions. The definition of the attributes is done in such a way that for each solution visited recently, the tabu list contains a corresponding attribute. All moves to solutions characterized by these attributes are forbidden (tabu). In this way cycles smaller than a certain length *t*, where *t* usually grows with the length of the tabu list, will not occur.

Besides a tabu status, a so-called aspiration criterion is associated with each attribute. If a current move leading to a solution s' is tabu, then this move will be considered admissible if s' satisfies the aspiration criterion associated with the attribute of s'. For example, we may associate with each attribute a threshold k for the objective function and allow a move m to a solution s' if $Z(s') \le k$, even though m is tabu.

The following algorithm describes the general framework of tabu search.

Algorithm Tabu Search

```
1.  Choose an initial solution $s \in S$;
2.  $best := Z(s)$;
3.  $s^* := s$;
4.  Tabu-list $:= \varphi$;
    REPEAT /*generation loop */
5.      $Cand(s)$ := { $s' \in N(s) |$ the move from $s$ to $s'$ is not tabu OR $s'$ satisfies the aspiration
                criterion};
6.      Generate a solution $\bar{s} \in Cand(s)$;
7.      Update the tabu list;
8.      $s := \bar{s}$;
9.      IF $Z(s) < best$ THEN
    BEGIN
10.     $s^* := s$;
11.     $best := Z(s)$;
      UNTIL stop criterion;
    END
```

Figure 4. Tabu search structure

Different stopping criteria and procedures for updating the tabu list T can be developed. We also have the freedom to choose a method for generating a solution $\bar{s} \in Cand(s)$. A simple strategy is to choose the best possible $\bar{s}$ with respect to function Z:

$$Z(\bar{s}) = \min\{Z(s') \mid s' \in Cand(s)\} \qquad \dots (1)$$

However, this simple strategy can be much too time-consuming, since the cardinality of the set $Cand(s)$ may be very large. For these reasons we may restrict our choice to a subset $V \subseteq Cand(s)$:

$$Z(\bar{s}) = \min\{Z(s') \mid s' \in V\} \qquad \dots (2)$$

Usually the discrete optimization problem (1) or (2) is solved heuristically.

7. Heuristic method

It is well known that the computation can be reduced by using a heuristic to act as an upper bound on the optimal solution prior to the application of branch and bound, since our problem $1|S_f|C_{\max} + E^w_{\max}$ is Np-hard and hence the existence of a polynomial time algorithm for finding an optimal solution is unlikely.

Therefore, developing fast heuristic algorithms, yielding near optimal solution is of great interest.

$R_f = \dfrac{S_f + \sum_{j=1}^{n_f} p_{j_f}}{n_f}$ and order these batches in non-decreasing order of ration R_f (f = 1, ..., F), hence compute completion time C_{i_f}.

8. Computational experience

This section reports the results of computational test to assess the effectiveness heuristics algorithms. These algorithms are coded in Matlab R2009b and runs on a Pentium IV at 2.00 GHz, 2.92 GB computer.

Test problems with (10, 30, 50, 100, 200, 500, 1000, 2000, 5000) jobs and with (2,4,6) families were generated as follows: jobs are distributed uniformly across families so that each family contains $\lceil n/f \rceil$ or $\lfloor n/f \rfloor$ jobs.

The processing time has been observed in the literature (e.g. [1]) that problem hardness is related to two parameters RDD and LF, called the relative range of due dates and the average lateness factor, respectively. In our experiment, RDD = 0.2, 0.4, 0.6, 0.8, 1.0 and LF = 0.2, 0.4 are used. Corresponding to each of these 5 × 2 = 10 cases, one problem instance is generated by selecting integer due dates $d_j, j \in N = \{1, 2, ..., n\}$, from interval [(1 – LF – RDD / 2) *SP*, (1 – LF + RDD / 2) *SP*], where $SP = \sum_{j \in N} p_j$. Sizes n = 10, 30, 50, 100, 200, 500, 1000, 2000 and 5000 are chosen.

The setup times are randomly generated integers from uniform distribution defined on [1,10]. Since the size of setup time's relation to processing times may affect problem "hardness", we generated problems with small (*S*), medium (*M*) and large (*L*) setup times. Medium setup times are randomly generated integers from the uniform distribution defined on [1,10]. Having generated an instance with small setup times $(S_f/2)$ and with large setup times $(2S_f)$ were constructed.

We generate problem for each contribution of n and setup times. Ten test problems created this method of data generation follows the one given in Hariri and Potts [10].

9. Comparative computational results

This section will report the results of our computational test to show the effectiveness for the local search methods (Memetic algorithms (MAs), Threshold acceptance method (TH) and Tabu search (TS)), we present tables of results which shows the importance of each of the methods. In each tables the first column gives the number of jobs the second column gives the number of families. The third column describes the average solution initial solution (Ini) which describe in section 3.1. The fourth, fifth and sixth columns describes the average computation for the local search MA, TH and TS respectively and the last three columns describes the average computation for the same local search but we started with the initial solution (Ini).

Table (1) Comparative results values for local search for $1|S_f/2|C_{\max}+E_{\max}^{w}$ problem with small setup

n	*S_f*	Ini	MA	TH	TS	MA+ Ini	TH+ Ini	TS+ Ini
10	2	173.6	127.4	139.7	125.3	124.2	132.9	124.6
	4	181.5	128.3	137.1	127.8	130	139.6	129.7
	6	182.9	132.6	142.1	132.3	131.9	135.6	131.9
30	2	781.2	476.5	603.9	937.5	453.2	597.1	557
	4	789.1	483.5	595	818.2	479.8	597.5	554.8
	6	791.7	487.3	610.5	761	461	617.2	563.5
50	2	1239.3	996	1046.5	1898.2	864.8	1038.1	994
	4	1238.2	964.1	1041.1	1945.6	860.2	1017.2	970.6
	6	1245	972.6	1021	1827.2	916	1026.5	968.5
100	2	2611.1	2660.8	2307.7	4588.6	1983.2	2330	2276.6
	4	2626.3	2770.8	2325.3	4653.7	1908.2	2327.4	2292.1
	6	2627.3	2777.9	2323.1	4643.9	2035.6	2313.5	2275.7
200	2	5484.6	7132.3	5256	10105.2	4674.5	5276.6	5239
	4	5606.3	7496.7	5339.4	10038.8	4758.8	5335.8	5306.9
	6	5650.6	7145	5379.3	10005.9	5046.4	5382	5364.7
500	2	13966.8	22188.2	13779.3	27603	13236.4	13779.9	13767.3
	4	14252	21984.4	14051.3	26525.9	13533.4	14064.6	14041.8
	6	14390	22008.9	14193.4	25750.5	13698.6	14208.8	14183.7
1000	2	28236.5	47770.9	28079.2	53649.2	27370	28078.3	28070.2
	4	28890.4	48747.1	28689.1	54508.4	28273.7	28689.3	28676.3
	6	29126.5	49111.6	28953.3	54641.7	28283.3	28958.2	28942.6
2000	2	57740	104370.6	57453.7	113057	56997.9	57457.7	57448.3
	4	59231	106132.4	58941.3	114372.1	58194.5	58936.7	58928.6
	6	59571.2	107192.1	59258.5	113988.6	58451.6	59262.1	59256.7
5000	2	144964.6	274167.4	144653.7	290780.8	144233.2	144647.1	144637.2
	4	147920.8	282533.9	147641.4	295423.8	147325.8	147635.1	147629.4
	6	148809.7	279114.8	148539.7	294655.7	148248.8	148546.3	148534.3

For the small setup times, the table (1) shows that TS with 10 jobs and the jobs (30, 50) with MA and TH with the large jobs all of them without using the initial solution gives the best values and (MA+Ini) with using initial solution gives the very best values for all test problems.

Table (2) Comparative results values for local search for $1|S_f|C_{\max}+E^{w}_{\max}$ problem with medium setup

n	S_f	Ini	MA	TH	TS	MA+ Ini	TH+ Ini	TS+ Ini
10	2	180.9	126.5	142.4	126.6	126.1	138.2	125.9
	4	196.1	133.9	152.5	137.2	134.4	152.3	135.4
	6	198.9	140.2	152.1	140.2	140.1	146.2	141.5
30	2	817.2	480.1	606.8	983.5	469.1	614.3	556.6
	4	828.7	517.8	621.8	692.2	443.8	625.1	566.3
	6	830.2	545.5	626.4	674.3	465.1	616.4	576.8
50	2	1254.8	1103.8	1045	1899.4	845.3	1039.3	964.3
	4	1255.3	971.5	1023.9	1969.8	856.2	1024.6	937.7
	6	1274.7	988.1	1020.9	1712.2	858.1	1030.8	953.3
100	2	2673.6	2737.9	2303.9	4509.4	1918.1	2326.2	2263.6
	4	2721.7	2710.4	2364.1	4521.1	2049.6	2361.1	2294.5
	6	2741.4	2732.3	2348.6	4726.5	2076.6	2326.4	2269.9
200	2	5664.7	7526.8	5326.7	10106.3	4568.6	5321.2	5311.3
	4	5894.9	7036.3	5513	10045.5	4770.3	5501.5	5465.5
	6	5993.5	6870.5	5613.3	10017.8	4878.2	5613.1	5553.4
500	2	14377.3	21730	14107.9	27702.2	13670.9	14109.2	14081.8
	4	14909.9	22229.3	14613.6	26531	13808	14630.5	14591.7
	6	15147	21347.7	14874.2	25986.6	14233.3	14863.9	14835
1000	2	29508.3	48695.1	29236	53596.5	28144.2	29237.3	29222.3
	4	30695.3	48737.3	30331.9	54719.7	29321.5	30336.1	30308.5
	6	31149.9	48606.6	30870	53966.5	29814.5	30849.2	30826.5
2000	2	60469.5	103838.9	60060.9	113059.3	59045.3	60058.8	60027.4
	4	63074.9	105811.6	62650	114354	61464.9	62651.7	62622.1
	6	63697.3	104952.8	63270.8	114000.1	61760.4	63263.8	63258.5
5000	2	150984.8	279512.1	150526.7	290782.4	149663	150551.3	150503.7
	4	156327	280041	155937.2	295409.4	154879.6	155912.5	155904.4
	6	158059.6	279012.8	157681.9	294665.4	156812.3	157668.4	157660.3

For the medium setup times, the table (2) shows that MA with the small jobs and TH with the large jobs together without using the initial solution gives the best values and (MA+Ini) with using initial solution gives the very best values for all tests.

Table (3) Comparative results values for local search for $1\left|2S_f\right|C_{\max}+E_{\max}^{w}$ problem with large setup

n	S_f	Ini	MA	TH	TS	MA+ Ini	TH+ Ini	TS+ Ini
10	2	200.7	130.1	151.5	133.9	128.1	148.2	131.5
	4	227.9	147.1	158.6	154.9	145.9	161.6	144.6
	6	234.5	162.5	174.3	162.5	163.6	175.2	163.9
30	2	896.3	476.6	994.6	987.8	464.8	667.5	584.9
	4	919.3	493	709.3	594.9	486.1	678.2	644.9
	6	924.5	521.1	707.7	586.1	508.3	707.2	656.8
50	2	1311.4	1030.3	1912	1902.4	840.9	1072.2	984.7
	4	1384.6	959.6	1982	1973.5	928.5	1107.7	1012.9
	6	1416	1125.3	1508.3	1480.8	921.1	1152	1038.6
100	2	2815.1	2639	4511.7	4511.7	1958.3	2352.9	2277
	4	3009.4	2809.7	4528.6	4528.6	2044.7	2505	2381.7
	6	3058.4	2660.6	4739.3	4739.3	2150.4	2535.9	2418.5
200	2	6077	6973.3	10109.3	10109.3	4869.7	5568.4	5555
	4	6612.9	6972.5	10060.8	10060.8	5273.2	5987.7	5865.6
	6	6810.1	6998.7	10044.6	10044.6	5437	6176.6	6080
500	2	15319.5	21738.8	27705.3	27705.3	14206.8	14895.7	14812.9
	4	16384	22671.3	26542.4	26542.4	14796.1	15918.7	15814.3
	6	16965.2	22146.6	26008.9	26008.9	15534.6	16541.2	16430.3
1000	2	32253.8	47365.8	53601.2	53601.2	29939.8	31754.4	31737.5
	4	34627.8	49374	54734	54734	31965.4	34037.9	33949.6
	6	35537	49680.8	53990.2	53990.2	33075.3	35017.6	34949.3
2000	2	66270.5	104653.1	113064.1	113064.1	63136.7	65603.1	65545.3
	4	71591.9	105038.8	114369.5	114369.5	68902.2	70895.5	70867.1
	6	72847.5	105937.1	114025.7	114025.7	70023	72177.9	72150.7
5000	2	164386.8	280971.6	290786	290786	161532.6	163661.4	163575.6
	4	175075.2	283666.4	295422	295422	172155.6	174451.2	174381
	6	178540.4	281541.6	294687	294687	176169.4	177902.8	177880.8

For the large setup times, the table (3) shows that MA without using the initial solution gives best values and (MA+Ini) with using initial solution gives the very best values for all tests.

Table (4) Comparative results times for local search for $1|S_f/2|C_{\max}+E_{\max}^{w}$ problem with small setup

n	S_f	MA	TH	TS	MA+ Ini	TH+ Ini	TS+ Ini
10	2	0.10883252	0.0137307	0.0403497	0.1105387	0.0137699	0.0144797
	4	0.11048101	0.0138458	0.012466	0.1114002	0.0137314	0.012437
	6	0.11036361	0.0133961	0.0135027	0.1103316	0.0137228	0.0131068
30	2	0.22324126	0.0139976	0.1915203	0.2205693	0.0139399	0.0154965
	4	0.22740866	0.0140644	0.029107	0.2381176	0.0142483	0.0135868
	6	0.21863862	0.0148078	0.0133969	0.2230498	0.0140883	0.0140049
50	2	0.34645271	0.0145218	0.3184211	0.331147	0.0146659	0.0196939
	4	0.34149563	0.014797	0.0996897	0.3636968	0.0151488	0.0148629
	6	0.32912527	0.0145969	0.0744137	0.3389998	0.0149821	0.0176986
100	2	0.64924318	0.0170516	0.5841508	0.6247026	0.0172662	0.0284134
	4	0.63836089	0.0169497	0.1807206	0.6898889	0.0171697	0.0192134
	6	0.62831271	0.0170007	0.1685608	0.6349765	0.0173874	0.0266064
200	2	1.31308241	0.0205188	0.7749002	1.2732417	0.0207114	0.0564775
	4	1.27441197	0.0212383	0.3445733	1.392774	0.0210271	0.0233039
	6	1.25789829	0.0210567	0.3043784	1.2986465	0.0211	0.0340326
500	2	3.9258198	0.0330525	1.7029673	3.914724	0.0328222	0.1559675
	4	3.91361761	0.0345531	1.0548164	4.1727172	0.0337647	0.0503596
	6	3.91102405	0.0340997	1.0233085	4.0015693	0.0346768	0.0835088
1000	2	11.129667	0.0531074	3.1141879	10.426894	0.053756	0.4873669
	4	11.1068385	0.0541592	2.4239156	11.02323	0.0546023	0.0776813
	6	10.3818238	0.0542381	2.3411212	10.661138	0.0540688	0.1440078
2000	2	30.9140773	0.0923056	6.2917285	30.560293	0.0916355	0.4479757
	4	30.9976885	0.0950825	4.7600457	30.118925	0.094348	0.2598253
	6	30.5013775	0.0950872	4.7795274	31.679902	0.0950037	0.4940084
5000	2	143.213298	0.2085531	16.812922	142.651	0.2114291	0.5326033
	4	143.165858	0.2170247	12.406821	142.32892	0.2172664	0.6975278
	6	143.269384	0.2196926	12.633429	143.40932	0.2174381	1.14055

For the small setup times, the table (4) shows that the TH without using the initial solution gives the best times and (TS+Ini) with the small jobs and (TH+Ini) with the large jobs together without using the initial solution gives best values.

Table (5) Comparative results times for local search for $1|S_f|C_{\max}+E^{w}_{\max}$ problem with medium setup

n	S_f	MA	TH	TS	MA+ Ini	TH+ Ini	TS+ Ini
	2	0.1115604	0.0136401	0.0159931	0.1129919	0.0137395	0.0167597
10	4	0.1095054	0.0137626	0.0127227	0.1109436	0.013559	0.0131628
	6	0.1085479	0.0138681	0.0140453	0.1102089	0.0137649	0.0129786
	2	0.2299197	0.0140374	0.0636041	0.2210766	0.0136717	0.0151958
30	4	0.2217523	0.0137109	0.0229393	0.2202173	0.0140356	0.0138933
	6	0.2240548	0.0141035	0.0140729	0.2207639	0.0137503	0.0142874
	2	0.3476012	0.0145405	0.1100502	0.3332888	0.0148581	0.0196615
50	4	0.3377080	0.0151462	0.1004303	0.3316289	0.0146339	0.0140731
	6	0.3332837	0.0148276	0.0709024	0.3310118	0.0150664	0.0144934
	2	0.6241669	0.0164551	0.1517987	0.6308705	0.016684	0.0262763
100	4	0.6318990	0.0173885	0.1816528	0.6268614	0.0171571	0.0175653
	6	0.6361715	0.0166663	0.1622131	0.6221015	0.0169756	0.0224208
	2	1.2919197	0.0206132	0.2966748	1.2653245	0.0205798	0.0548671
200	4	1.2858324	0.0213033	0.3422599	1.2724934	0.0213418	0.0229836
	6	1.2601809	0.0206085	0.3308933	1.2696572	0.02102	0.0336048
	2	3.9914567	0.034162	1.0523205	3.8947271	0.0331786	0.1456234
500	4	4.0022477	0.034393	1.1051942	3.9464552	0.0347637	0.0442465
	6	3.8958152	0.0338068	1.1845936	3.912345	0.0338649	0.0761994
	2	10.557029	0.0534388	2.3461806	10.392622	0.0525992	0.4698457
1000	4	10.492383	0.055572	2.4322482	10.400127	0.0542152	0.0733618
	6	10.385844	0.0546621	2.3666651	10.404557	0.0538453	0.138249
	2	30.742108	0.0913924	4.8846509	30.265746	0.0926708	0.3943389
2000	4	30.488132	0.0950017	4.9332271	30.028588	0.0940821	0.2410351
	6	30.475221	0.0946941	4.7403941	30.311575	0.0955163	0.396691
	2	142.77779	0.2110201	12.368691	141.82468	0.2109633	0.5856263
5000	4	142.11019	0.216591	12.531411	142.58962	0.2192247	0.6429186
	6	141.35881	0.2181644	12.714973	144.18198	0.2185565	1.1054644

For the medium setup times, the table (5) shows that TH without using the initial solution and (TH+Ini) with using initial solution gives the best times. And with small jobs a few numbers for testing TS and (TS+Ini) gives good times.

Table (6) Comparative results times for local search for $1|2S_f|C_{\max}+E^w_{\max}$ problem with large setup

n	*S_f*	MA	TH	TS	MA+ Ini	TH+ Ini	TS+ Ini
10	2	0.10896699	0.0135174	0.0178549	0.1106679	0.0137978	0.0183556
	4	0.10860462	0.0133129	0.0127746	0.115008	0.0142408	0.0132971
	6	0.10925139	0.0136384	0.0134243	0.1104732	0.0135868	0.0133486
30	2	0.21915571	0.0140236	0.0632078	0.2207647	0.014408	0.017724
	4	0.21369427	0.0139407	0.017807	0.2244495	0.013773	0.0142773
	6	0.21910201	0.0142965	0.0148293	0.2194332	0.014182	0.0142825
50	2	0.32694722	0.0145657	0.108624	0.3352672	0.0146524	0.0187651
	4	0.32840375	0.0144344	0.1003529	0.3394235	0.0159645	0.0148319
	6	0.33370958	0.0141846	0.0528694	0.3328943	0.0148509	0.0144555
100	2	0.63208899	0.0160474	0.152148	0.6338799	0.016436	0.0245161
	4	0.6214911	0.0163935	0.1778174	0.6389653	0.0171848	0.0171098
	6	0.62751383	0.0159664	0.1577434	0.6268194	0.0167408	0.0178948
200	2	1.2656667	0.0190432	0.3192943	1.2946297	0.0204723	0.0542499
	4	1.25827895	0.0184829	0.3479979	1.3080298	0.0212982	0.0231382
	6	1.25143191	0.0191695	0.3155637	1.2700352	0.0212266	0.0275675
500	2	3.896433	0.0289519	1.0648652	3.9246129	0.0330187	0.158508
	4	3.90758209	0.0295286	1.0575683	3.9801845	0.0342477	0.0446724
	6	3.88707638	0.0291678	1.0301988	3.9109548	0.0347245	0.0725726
1000	2	10.380885	0.0452365	2.4232781	10.301447	0.0531206	0.4842858
	4	10.5380927	0.0445134	2.4630021	11.615705	0.0542899	0.0724215
	6	10.4861636	0.044968	2.4022801	10.394567	0.0546751	0.1303602
2000	2	30.3273213	0.0746369	4.8613714	29.623444	0.092835	0.4144581
	4	30.3472568	0.0762397	4.7586922	30.109954	0.0946109	0.217272
	6	30.6621402	0.0760138	4.6319009	30.305964	0.0958078	0.4695093
5000	2	140.225481	0.1684806	12.40045	139.21983	0.2107117	0.5875704
	4	141.421382	0.1712416	12.216357	140.99398	0.215821	0.6082168
	6	143.013632	0.1685316	13.039484	140.95291	0.2178962	1.0377402

For the large setup times, the table (6) shows that TH without using the initial solution and (TH+Ini) with using initial solution gives the best times. And with a few numbers for testing TS and (TS+Ini) gives good times.

10. Conclusions

In this paper, we have developed a number of solution procedures for the single machine scheduling problem:

Minimize the maximum completion time and maximum weighted earliness time $C_{\max}+E^w_{\max}$ taking into account sequence with setup times. The local search methods that are used to solve all of the large problems in this paper, the results show the robustness and flexibility of local search heuristics.

Future work Some suggestions for future research are described as follows:

First, the propose of extension the exact for $1|S_f|C_{\max}+E^w_{\max}$ problem by driving a good lower bound or using the dominance rule in branch and bound algorithm.

Second, using the local search heuristic should be explored finding an improvement potential of various polynomially bounded scheduling heuristic.

Reference

1. Abdul-Muhsin V., “*Exact and near optimal solutions for a single machine scheduling problem to minimize the weighted sum of squares completion time*”, M. Sc., College of Science, University of Al-Mustansiriyah, (2001).
2. Abdul-Razaq T.S. and Potts C.N., "*Dynamic Programming Stat-Space Relaxation for Single Machine Scheduling*", Journal of Operations Research Society, 39, 141-152, (1988).
3. Allahverdi A., Gupta J.N.D. and Aldowaisan T., "*A Review of Scheduling Research Involving Setup Considerations*", Omega, in T. Mgmt sci. 27, 219-239, (1999).
4. Bruno J. and Downey P. "*Complexity of Task Sequencing with Deadlines, Setup Times and Changeover Cost*", SIAMJ Compute 7, 393-404, (1978).
5. Buriol L., França P. M. and Moscato, P., "*Recursive Arc Insertion: A New Local Search Embedded in a Memetic Algorithm for the Asymmetric Traveling Salesman Problem*", Submitted to Journal of Heuristics, (1999).
6. Chen B.,"*A Better Heuristic for Preemptive Parallel Machine Scheduling with Batch Setup Times*", SIAMJ Computer 22, 1303-1318, (1993).
7. França P. M., Mendes A. and Moscato, P., "*A Memetic Algorithm for the Total Tardiness Single Machine Scheduling Problem*", European Journal of Operational Research (to appear) (1999) .
8. Gendreau M., An introduction to tabu search. In Glover F. and Kochenberger G.A., editors, "*Handbook of metaheuristics*", 37-54. Boston: Kluwer Academic Publishers, (2003).
9. Glover F. and Laguna M., "*Tabu search*”. Boston: Kluwer Academic Publishers, (1997).
10. Hariri A.M.A. and Potts C.N., "*Single Machine Scheduling with Batch Set-up Times to Minimize Maximum Lateness*", Annals of Operations Research 70, 75–92, (1997).
11. Moghaddam R.T., Moslehi G., Vasei M. and Azaron A., "*Optimal Scheduling for a Single Machine to Minimize the Sum of Maximum Earliness and Tardiness Considering Idle Insert*”, Applied Mathematics and Computation, 167, 1430-1450, (2005).
12. Monma C.L. and Potts C.N., "*The Complexity of Scheduling with Batch Setup Times*", Operations Research 37, 798-804, (1989).
13. Moscato P., "*On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*", Caltech Concurrent Computation Program, C3P Report 826, (1989).
14. Moscato P., "*Memetic Algorithms*: *a Short Introduction*". In Corne D., Dorigo M., and Glover F. (eds.) New Ideas in Optimization (McGraw-Hill) (1999).
15. Van-Wassenhove L.N. and Gelders F., "*Solving a Bicriterion Scheduling Problem*". European Journal of Operations Research, 4, 42-48, (1980).