Algorithms for Scheduling a Single Machine to Minimize Total Completion Time and Total Tardiness

Tariq S. Abdul-Razaq and Faez H. Ali
Math. Dept, College of Sciences, University of Al-Mustansiriya, Baghdad, Iraq.
faezhasan@yahoo,com

Abstract

In this paper we look at the problem where we have to schedule n jobs with processing times and due dates on a single machine. The objective is to find a schedule that minimize a function of the sum of completion time and sum of tardiness (i,e to minimize the multiple objective functions (ΣC_i,ΣT_i)). We propose two methods for solving this simultaneous minimization problem to find the set of all efficient solutions, (Pareto optimal solutions). This set of all efficient solutions is not easy to find, therefore, it could be preferable to have an approximation to that set in a reasonable amount of time. Therefore branch and bound (BAB) and local search methods are used.

The Particle Swarm Optimization (PSO) method is applied as new local search method on a set of randomly generated problems to solve machine scheduling problem with multiple objective functions. Comparison studies are made between Branch and Bound Methods (BAB), PSO and Genetic Algorithm (GA) to show which one is the better method in applications. In addition, tuning the parameters of every method has been suggested in order to improve the application of every method. A new style of development steps has been proposed to achieve good convergence in application. Since our problem is NP-hard, we propose new heuristic method like PSO and GA to find approximation solutions especially when the number of jobs exceeds the ability of some exact methods like complete enumeration and BAB in solving such problems.

Lastly, the proposed methods results are compared for this multi-objective scheduling problem. Computational experience is found that these local search algorithms solve problem to '2000 'jobs with reasonable time.

Key words: Multiple objective Scheduling, Branch and Bound, Genetic Algorithm, Particle Swarm Optimization.

1. Introduction

1.1 Terminology

The function to be maximized or minimized with or without subject to certain constraints is called the **objective function**. A schedule **σ** for the minimum problem is said to be **feasible** if it satisfies the corresponding constraints. The set of all feasible schedules are called the **set of solutions**.

For many years, scheduling researches focused on single (objective) performance measures. In most real world, a scheduling application, with more than one performance measure is of interest. The multi-criteria (multi-objective) have received significant attention in recent years (Nagar et al. 1995). The multi-criteria scheduling problem can be stated as follows. There are n jobs to be processed on a single machine, each job i has processing time p_i and due date d_i at which ideally should be completed. Penalties are incurred whenever a job i is completed earlier or later than its due date d_i. Multi-criteria optimization with conflicting objective functions provides a set of Pareto optimal solutions, rather than one optimal solution. This set includes the solution that no other solution is better than with respect to all objective functions. In the literature, there are two approaches for multi-criteria scheduling problems: the **hierarchical approach** and the **simultaneous approach**. In the hierarchical approach, one of the two criteria is considered as the primary criterion and the other one is considered as the secondary criterion. The problem is to minimize the primary criterion while breaking ties in favor of the schedule has minimum secondary criterion value. The studies by Chang and Su (2001) (1) and Chen and Qi (1997) (2) are examples of hierarchical minimization problem with earliness and tardiness costs.

For the simultaneous approach, there are two types; the first one is to find the sum of these objectives. The second one typically generates all efficient schedules (set of **Pareto optimal solutions**) and selects the one that yields the best composite objective function value of the criteria. Several studies by Van Wessenhove and Gelder (1980) (3), Hoogeveen J. (1996) (4), Alasaf (2007) (5), Findi (2012) (6) and Hoogeveen J. (2005) (7) are examples of simultaneous minimization scheduling problems.

This research effort proposes two different approaches to find efficient (Pareto optimal solutions) for $1//(\Sigma C_i,\Sigma T_i)$ problem by using **Branch and Bound (BAB)** method from one side and, **Particle Swarm Optimization (PSO)** and **Genetic Algorithm (GA)** as two new local search methods from the other side, for multi-criteria scheduling problem.

The meaning of the **Particle Swarm Optimization (PSO)** refers to a relatively new family of algorithms that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems. PSO is an extremely simple algorithm that seems to be effective for optimizing a wide range of Applications (7).

Genetic Algorithms (GA's) are a class of optimization algorithms. GA's attempt to solve problems through modeling a simplified version of genetic process. There are many problems for which a GA approach is useful. It is, however, untraditional if assignment is such a problem (8).

This problem, like all deterministic scheduling problems belongs to class of simultaneous optimization, which are well known to be NP-hard since the $1//\Sigma T_i$ is NP-hard (9).

1.2 Notations and Basic Concepts

The following notation will be used in this paper:

n	:	number of jobs
p_i	:	processing time of job i
d_i	:	due date of job i
C_i	:	completion time of job i
T_i	:	the tardiness of job i
SPT	:	shortest processing time
EDD	:	Earliest due date
BAB	:	Branch and Bound
MSP	:	machine scheduling problem
MOF	:	multi-objective function
pbest	:	previous best position

The following sequencing rules and basic concepts are used in this paper:

SPT rule: Jobs are sequencing in non-decreasing order of p_i, this SPT (shortest processing time) rule is used to minimize ΣC_i for $1//\Sigma C_i$ problem (10).

EDD rule: Jobs are sequencing in non-decreasing order of d_i, this EDD (Earliest due date) rule is used to minimize T_{max} for $1//T_{max}$ (11).

Definition (11): The term **optimize** in a multi-criteria decision making problem refers to a solution around which there is no way of improving any objective without worsening the other objective.

Definition [12]: A feasible schedule σ is **Pareto optimal** (PO), or non-dominated (efficient) with respect to the performance criteria f and g if there is no feasible schedule π such that both $\mathbf{f(\pi)\le f(\sigma)}$ **and** $\mathbf{g(\pi)\le g(\sigma)}$, where at least one of the inequalities is strict.

Emmon's Theorem [2]: For the $1/ /\Sigma T_i$ problem, if $\mathbf{p_i\le p_j}$ **and** $\mathbf{d_i\le d_j}$ then there exists an optimal sequencing in which job i sequencing before job j.

Al-Magraby's Lemma [2]: For the $1/ /\Sigma T_i$ problem, if $d_j\ge\sum_{i=1}^{n} p_i$, then there exists an optimal sequence in which job j sequencing last.

Smith Backward Algorithm (SBA) [2]

This algorithm is used to solve $1/\bar{d}_i/\Sigma C_i$ problem, the main steps of this algorithm:

Step(1): set $t=\sum_{j\in N}^{n} p_j$, k=n, N={1,…,n}.

Step(2): Find a job $j\in N$ such that

(1) $\bar{d}_j\ge t$, (2) $p_j\ge p_i$ for each $i\in N$ and $\bar{d}_j\ge t$, then assign job j in position k.

Step(3): Set $t=t-p_j$, N=N-{j} and k=k-1, if k>1 goto step(2), otherwise stop.

The organization of this paper is as follows: in section two we present multiple objective problems. Section three and four present the mathematical model and discuss the BAB, PSO and GA methods. Implementation, experimental results, analysis and conclusions are given in the last sections.

1. Multiple Objective Problems

The **Machine Scheduling Problems (MSP)** plays a very important role in most manufacturing and production systems as well as in most information processing environment. Scheduling theory has been developed to solve problems occurring in for instance production facilities. The basic scheduling problem can be described as finding for each of the tasks, which are also called jobs, an execution interval on one of the machines that are able to execute it, such that all side-constraints are met; obviously, this should be done in such a way that the resulting solution, which is called a schedule, is best possible, that is, it minimizes the given objective function [13].

For many years, scheduling researchers focused on single regular performance measures that are non-decreasing in job completion time.

Typically, each criterion has been studied separately, even though most real life scheduling problems involve multiple criteria [14]. However few studies considered multiple criteria together. Three types of multiple criteria problem can be identified. The first of these types of problems involves identifying all sequence that minimizes the first objective. One of these sequences that minimize a second objective is chosen as the optimal sequence for that problem this approach is called hierarchical approach. The second of these multiple criteria problems, when the criteria are weighted differently, an objective functions and transform the problem into a single criterion scheduling problem. This approach is called simultaneous optimization along with the third type of multiple criteria problems [15].

The third one of these multiple criteria problems is going to consider both criteria as equally important. The problem now is to find a sequence that does well on both objectives.

Scheduling problem is specific case of the multiple objective (multi-criteria) scheduling problems can be formulated as follows: minimize or maximize $F(s)=(f_1(s),(f_2(s),\ldots,f_k(s))$ s.t. $s\in S$ where s is a solution, S is the set of feasible solutions, k is number of objectives in the problem, F(s) is the image of s in the k-objective space and each $f_i(s)$, i=1,…,k represents one (minimization or maximization) objective.

In many problems, the aim is to obtain the optimal arrangement of group of discrete entities in such a way that the additional requirements and constraints (if they exist) are satisfied. If the problem is a multi-objective one, various criteria exist to evaluate the equality of solution and there is an objective (Min. or Max.) attached to each of these criteria (15).

The literature on multiple objective problems for single machine problems is summarized by Dileepan and Sen (1988) (16), Fry et al. (1989) (17), Hoogeveen J. (1992) (18), Lee and Vairaktarakis (1993) (19) and Nagar et al. (1995) (20) provide a detailed for MSP's.

3. Mathematical Model and Analysis

The problem of scheduling N={1,2,…,n} the set of n jobs which are processed on a single machine to minimize the multi-criteria may be stated as follows. Each job $i\in N$ has is to be processed on a single machine which can handle only one job at a time, job i has a processing time p_i and due date d_i, all jobs are available for processing at a time zero.

If a schedule $\sigma=(1,2,\ldots,n)$ is given, then the earliest completion time $C_i=\sum_{j=1}^{i} p_j$ for each job i can be computed and consequently the tardiness of job i $T_i=\max\{C_i-d_i,0\}$ is easy to compute. Our objective is to find a schedule $\sigma\in S$ (where S is the set of all feasible schedule) that minimizes the multi-criteria $(\Sigma C_i,\Sigma T_i)$ for the $1//(\Sigma C_i,\Sigma T_i)$ problem.

This problem belongs to simultaneous optimization and written as:

$$\begin{aligned}&\text{Min } \{\Sigma C_i , \Sigma T_i\}\\ &\text{Subject to}\\ &\left.\begin{aligned}&C_i \ge p_i, && i=1,2,\ldots,n.\\ &T_i \ge C_i-d_i, && i=1,2,\ldots,n.\\ &T_i \ge 0, && i=1,2,\ldots,n.\end{aligned}\right\} \qquad \ldots(P_1)\end{aligned}$$

It's clear that there are two special cases for the problem (P_1). The first one is $1//Lex(\Sigma C_i , \Sigma T_i)$ problem which can be written as:

$$\begin{aligned}&\text{Min } \{\Sigma T_i\}\\ &\text{Subject to}\\ &\left.\begin{aligned}&\sum_{i=1}^{n} C_i = \Delta,\\ &\text{where } \Delta=\sum_{i=1}^{n} C_i(SPT)\end{aligned}\right\} \qquad \ldots(P_2)\end{aligned}$$

Its well-known for the problem (P_2), the object ΣC_i is more important than ΣT_i since the multi-criteria object is $Lex(\Sigma C_i ,\Sigma T_i)$. It's clearly a feasible schedule for (P_2) is obtained by SPT rule in which ΣC_i is optimal.

The only chance to minimize ΣT_i is to use the special cases for the jobs with the same processing times (see section 3.1).

The second one is $1//\Sigma C_i + \Sigma T_i$ problem which can be written as:

$$\begin{aligned}&\text{Min } \{\Sigma C_i + \Sigma T_i\}\\ &\text{Subject to}\\ &\left.\begin{aligned}&C_i \ge p_i, && i=1,2,\ldots,n.\\ &T_i \ge C_i-d_i, && i=1,2,\ldots,n.\\ &T_i \ge 0, && i=1,2,\ldots,n.\end{aligned}\right\} \qquad \ldots(P_3)\end{aligned}$$

The aim for the problem (P_3) is to find a processing order of the jobs on a single machine to minimize the sum of total completion times and the total tardiness, which is a single object and can be minimized by BAB method.

For multi-criteria, if the objectives can be optimized individually, then we can deduce that the set of efficient solutions have no more elements only one with extreme values of the individul objective functions. The above fact can be seen in the following special cases:

Case (1): A schedule σ obtained by ordering the jobs in a non-decreasing order of thier processing times (SPT-rule) is optimal for both objectives ($\Sigma C_i,\Sigma T_i$) if $d_{\sigma(i)}+p_{\sigma(i)}\leq C_{\sigma(i+1)}$ for all i=1,2,…,n-1.

Case (2): From Emman's theorem, if the SPT and EDD rules are identical then there exist **only one** effeceint solution for (P_1).

Case (3): If $p_i=p$, ∀i, p is positive integer and a schedule σ obtained by ordering the jobs in a non-decreasing order of due dates (EDD-rule) is optimal for both objectives ($\Sigma C_i,\Sigma T_i$).

Case (4): If $d_i=d$, ∀i, d is positive integer and a schedule σ obtained by ordering the jobs in a non-decreasing order of processing times (SPT-rule) is optimal for both objectives ($\Sigma C_i,\Sigma T_i$).

Note that case (3) and case (4) are special cases of case (2).

Case (5): From Al-Magrapy lemma, if $d_j\geq \sum_{i=1}^{n} p_i$, and $p_j=\max_{i\in N}\{p_i\}$, and this also satisfies for each job k∈N-{j}, then there exists **only one** efficient solution for (P_1).

Proposition (1):

There exists an efficient solution for problem (P_1) that satisfies the SPT rule.

Proof:

1. Suppose first, that all processing times are different. The unique SPT sequence (SPT^*) gives the absolute minimum of ΣC_i. Hence there is no sequence σ≠ SPT^* such that:

$$\sum_{i=1}^{n} C_i(\sigma)\leq\sum_{i=1}^{n} C_i(SPT^*) \quad \text{and}$$

$$\sum_{i=1}^{n} T_i(\sigma)\leq\sum_{i=1}^{n} T_i(SPT^*) \qquad \dots(1)$$

with at least one strict inequality.

2. If more than one SPT sequence exists (jobs with equal processing times), let SPT^* be a sequence satisfying the SPT rule and the jobs with equal processing times are ordered in EDD rule satisfy the special case (2) above to minimize $\Sigma T_i(SPT^*)$. Note that if we have SPT^* is not unique we can prove that every SPT^* sequence is an efficient, it is clear that sequence that do not satisfy the SPT rule cannot dominate an SPT^* sequence (1). Note if σ is an SPT but not SPT^* sequence it cannot dominate SPT^* since:

$$\sum_{i=1}^{n} C_i(\sigma)=\sum_{i=1}^{n} C_i(SPT^*) \text{ and}$$

$$\sum_{i=1}^{n} T_i(SPT^*)\leq\sum_{i=1}^{n} T_i(\sigma) \qquad \dots(2)$$

Hence all SPT^* sequences are efficient.

Proposition (2): If $T_{max}(EDD)=0$, then there exists an efficient sequence for 1/ /($\Sigma C_i,\Sigma T_i$) problem obtained by Smith backward algorithm (SBA).

Proof: If $T_{max}(EDD)=0$, then it's clear that SBA gives a schedule with $C_i\leq d_i$ for each i∈N and this schedule also gives minimum ΣC_i and with $\Sigma T_i=0$. This schedule cannot dominated by any other schedule since ΣC_i is minimum for all schedules with $\Sigma T_i=0$. Hence this schedule obtained by SBA is efficient for 1/ /($\Sigma C_i,\Sigma T_i$) problem.

Note that the purpose of any algorithm process is to find for each problem instance a feasible solution called optimal that minimize their objective function. This usual meaning of the optimum makes no sense in the multi-criteria case because it doesn't exist in most of the cases, a solution optimizing all objectives simultaneously.

Hence we search for feasible solutions yielding the best compromise among objectives that constitutes a so called **efficient solution set**.

4. Methods of Approach

There are approaches that can be used for solving multi-criteria scheduling problems, which are to find the set of efficient solutions or at least approximation to it. It is clear that this set of all efficient solutions is difficult to find. Therefore, it could be preferable to have an approximation to that set in a reasonable amount of time.

We will introduce two methods of approach to solve multi-criteria scheduling problem (P_1) for finding the set of efficient solutions.

4.1 Branch and Bound Method for (P_1)

This method, depend on the techniques of branch and bound (BAB) algorithm with some modifications. The BAB method is characterized by its branching procedure, upper and lower bounding procedures and search strategy.

We present a constructive BAB algorithm to find all or some of the efficient solutions (**Pareto optimal points** (**POP**)) when the two criteria ΣC_i and ΣT_i are of simultaneous interest in problem (P_1). The main idea of this BAB algorithm is depending on properties of BAB algorithm and some modifications such as using the definition of efficient solutions and without reset the upper bound (UB) at the last level of BAB method. The main steps of the BAB algorithm as follows:

Step(1): Find the proposed UB by SPT rule, that is sequencesing the job in non-decreasing order of their processing time p_i, i=1,2,...,n, for this order σ calculate $\Sigma C_i(\sigma)$ and $\Sigma T_i(\sigma)$ and set UB=($\Sigma C_i(\sigma)$,$\Sigma T_i(\sigma)$) at the parent node of the search tree. UB is efficient by proposition (1) and add this efficient solution to the set of POP. If T_{max}(EDD)=0, then there exists an efficient sequence obtained by proposition (2), and also add this efficient solution to the set of POP.

Step(2):For each partial sequence of jobs σ (i.e., for each node in the search tree), compute the lower bound LB(σ) as follows:
LB(σ)=exact cost of σ + cost of S' (where S' the set of unsequence jobs), obtained by sequence the jobs in SPT rule.

Step(3):Branch from each node with LB(σ) ≤ UB.

Step(4):At each node of the last level of the BAB method, if (ΣC_i,ΣT_i) denote the outcome, then add this outcome to the set of POP, unless it is dominated by the previously obtained POP.

Step(5): Stop.

4.2 Local Search Methods for (P_1)

Evolutionary Algorithms (EAs) [(21)] have been shown to be successful for a wide range of optimization problems. While these algorithms work well for many optimization problems in practice, a satisfying and rigorous mathematical understanding of their performance is an important challenge in the area of evolutionary computing [(22)].

4.2.1 Genetic Algorithms [(23)]

Genetic Algorithms (GA's) are search algorithms based on the mechanics of natural selection and natural genetics. GA is an iterative procedure, which maintains a constant size population of candidate solution. During each iteration step (Generation) the structures in the current population are evaluated, and, on the basic of those evaluations, a new population of candidate solutions formed. The basic GA cycle shown in figure (1).

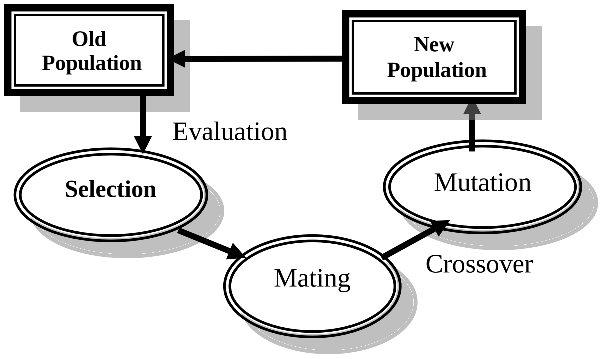

Figure (1) Basic cycle of GA.

An abstract view of the GA is:

```
Generation=0;
Initialize G(P); {G=Generation ; P=Population}
Evaluate G(P);
While (GA has not converged or terminated)
        Generation = Generation + 1;
        Select G(P) from G(P-1);
        Crossover G(P);
        Mutate G(P);
        Evaluate G(P);
End (While)
Terminate the GA.
```

4.2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) has found applications in a lot of areas. In general, all the application areas that the other evolutionary techniques are good at are good application areas for PSO [7].

PSO was originally developed by a social-psychologist J. Kennedy and an electrical engineer R. Eberhart in 1995 and emerged from earlier experiments with algorithms that modeled the flocking behavior seen in many species of birds. It is yet another optimization algorithm that falls under the soft computing umbrella that covers genetic and evolutionary computing algorithms as well [24].

PSO is an extremely simple concept, and can be implemented without complex data structure. No complex or costly mathematical functions are used, and it doesn't require a great amount of memory [25]. The facts of PSO has fast convergence, only a small number of control parameters, very simple computations, good performance, and the lack of derivative computations made it an attractive option for solving the problems.

The **PSO algorithm** depends in its implementation in the following two relations:

$$v_{id}=w*v_{id}+c_1*r_1*(p_{id}-x_{id})+c_2*r_2*(p_{gd}-x_{id}) \quad \ldots(3a)$$

$$x_{id} = x_{id} + v_{id} \quad \ldots(3b)$$

where c_1 and c_2 are positive constants, r_1 and r_2 are random function in the range [0,1], $x_i=(x_{i1},x_{i2},\ldots,x_{id})$ represents the ith particle; $pa_i=(p_{i1},p_{i2},\ldots,p_{id})$ represents the (pbest) best previous position (the position giving the best fitness value) of the ith particle; the symbol g represents the index of the best particle among all the particles in the population, $v=(v_{i1},v_{i2},\ldots,v_{id})$ represents the rate of the position change (velocity) for particle i [7].

The original procedure for implementing PSO is as follows:

1. Initialize a population of particles with random positions and velocities on d-dimensions in the problem space.
2. PSO operation includes:
   a. For each particle, evaluate the desired optimization fitness function in d variables.
   b. Compare particle's fitness evaluation with its pbest. If current value is better than pbest, then set pbest equal to the current value, and pa_i equals to the current location x_i.
   c. Identify the particle in the neighborhood with the best success so far, and assign it index to the variable g.
   d. Change the velocity and position of the particle according to equation (3a) and (3b).
3. Loop to step (2) until a criterion is met.

Like the other evolutionary algorithms, a PSO algorithm is a population based on search algorithm with random initialization, and there is an interaction among population members. Unlike the other evolutionary algorithms, in PSO, each particle flies through the solution space, and has the ability to remember its previous best position, survives from generation to another.

A number of factors will affect the performance of the PSO. These factors are called **PSO parameters**, these parameters are [24]:

1. Number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (less particles) must be sought.
2. Maximum velocity (v_{max}) parameter. This parameter limits the maximum jump that a particle can make in one step.
3. The role of the inertia weight w, in equation (3a), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one.
4. The parameters c_1 and c_2, in equation (3a), are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima, c_1 than a social parameter c_2 but with $c_1 + c_2 = 4$.
5. The parameters r_1 and r_2 are used to maintain the diversity of the population, and they are uniformly distributed in the range [0,1].

4.2.3 Analysis of Number of Efficient Solutions

As our main aim in this research is to identify the set of all efficient solutions, we should try to hold the entire set (i.e., the set of all efficient solutions).

It's clear from special cases of the problem (P_1) above (section 3.1), if the criteria (objectives) can be optimized individually, we can deduce the set of efficient solutions have only one element with extreme values of the individual objective functions. Since our algorithm depends on BAB method, we can sure that a solution is truly an efficient solution. We proved that the SPT rule is one the efficient solution, hence we can determine if some solutions of the BAB method are dominated by the SPT solution and other solutions. Also we proved that if $T_{max}(EDD)=0$, then SBA gives one of the efficient solution for problem (P_1).

4.3 Branch and Bound Method for Problem (P_3)

The main aim for problem (P_3) is to find a schedule σ of the jobs on a single machine to minimize $\Sigma C_{\sigma(i)}+\Sigma T_{\sigma(i)}$, $\sigma\in S$, where S is the set of all feasible solutions.

4.3.1 Derivation of Lower Bound for Problem (P_3)

Consider the formulation of the problem (P_3), the problem can be decomposed into two subproblems with a simple structure. Then the lower bound of the problem (P_3) is calculated as follows:

Consider the two subproblems (SP_1) and (SP_2) as follows:

$$Z_1 = \min_{\sigma\in S}\{\sum_{j=1}^{n} C_{\sigma(j)}\}$$

$$\left.\begin{array}{ll} \text{s.t.} & \\ C_{\sigma(j)} \ge p_{\sigma(j)}, & j=1,2,\dots,n. \\ C_{\sigma(j)} \ge C_{\sigma(j-1)} + p_{\sigma(j)}, & j=2,3,\dots,n. \end{array}\right\} \quad \dots(SP_1)$$

$$Z_2 = \min_{\sigma\in S}\{\sum_{j=1}^{n} T_{\sigma(j)}\}$$

$$\left.\begin{array}{ll} \text{s.t.} & \\ T_{\sigma(j)} = C_{\sigma(j)} - d_{\sigma(j)}, & j=1,2,\dots,n. \\ T_{\sigma(j)} \ge 0, & j=1,2,\dots,n. \end{array}\right\} \quad \dots(SP_2)$$

Its clear that for the decomposition, (SP_1) and (SP_2) have simpler structure than (P_3), and thus appear easily first to solve optimality for (SP_1) to get Z_1 by applying shortest processing time (SPT) rule. Second, to get a lower bound for (SP_2), let σ be the sequence jobs and S' be the set of unsequence jobs.

Hence

$$LB(\sigma) = \sum_{i\in\sigma} T_i(\sigma) + \sum_{i\in S'} T_i(S')$$

Where $\sum_{i\in\sigma} T_i(\sigma)$ exact cost of σ and $\sum_{i\in S'} T_i(S')$ is obtained by using lower bound methods.

If we ignore the cost(S'), we get a weak lower bound for (SP_2).

Now calculate Z_1 to be the minimum value for (SP_1) and $LB(\sigma)$ to be the lower bound for (SP_2), then applying the following result:

Theorem (1) (Hoogeveen H., 2005):

$Z_1+LB(\sigma) \le Z$, where Z_1 is the minimum objective function value of (SP_1), $LB(\sigma)$ is a lower bound for (SP_2) and Z is the minimum objective value of (P_3).

4.3.2 Derivation of Upper Bound for Problem (P_3)

We propose to use a simple heuristic solution which is obtained by ordering the jobs in SPT rule to provide an initial upper bound (UB) on the MOF. Let σ, $\sigma=(\sigma(1),\sigma(2),...,\sigma(n))$ be such ordered, then:

$$UB = \sum_{j=1}^{n} C_{\sigma(j)} + \sum_{j=1}^{n} T_{\sigma(j)} \qquad \ldots(4)$$

5. Implementation of Local Search Methods for (P_1)

Obviously the problems including more than one criterion are more difficult. So there is a need for local search methods to treat a large size instances problem. This is the main aim of the present paper.

Effectively, evolving methods or can be called Local Search methods like PSO and GA have demonstrated their ability to solve multi objective problems to find the approximation set of efficient solutions for the problem (P_1).

In this section, we are going to describe the two methods of local search. The first is the PSO as the main new method, and the second, is GA as a comparative method to compare the results obtained from the two methods in order to find which is better.

Before we discuss each of the methods, we have to talk about the common basics between the two methods, these basics are:

1. **Problem Definition**
   The most prominent member of the rich set of combinatorial optimization problems is undoubtedly the Machine Scheduling Problem (MSP). In order to find the set of POP, we solve the problem (P_1) of minimizing $(\Sigma C_i,\Sigma T_i)$. Obviously, this scheduling problem is example of NP-complete, the work area to be explored grows exponentially according with number of jobs, and so does. In general, if n jobs were must be arranged in a single machine, then the general complexity is n!.
2. **Problem Representation**
   The solution representation should be an integer vector. In this particular approach we accept schedule representation which is described as a sequence of jobs.
3. **Initial Population**
   For the initialization process we can either use some heuristics starting from different jobs, or we can initialize the population by a random sample of permutation of $N=\{1,2,\ldots,n\}$.

5.1 Use of GA in MSP

Now we will discuss the use of GA first, since it has been used before in MSP for many times.

1. **Genetic Operators**

- **Selection Operator**
  This method uses the **roulette wheel** selection method. The sequence with low fitness has a higher probability of contributing one or more offspring to the next generation.
- **Crossover Operator**
  The strength of genetic algorithms arises from the structured information exchange of crossover combinations of highly fit individuals. So what we need is a crossover-like operator that would exploit important similarities between chromosomes. For that purpose the crossover used in this algorithm is the **Order Crossover** (OX) [26], this operator chooses two random crossover points, for example, if the parents are:

$$\begin{array}{ll|lll|lll|lll} v_1 & : & 7 & 9 & 8 & 2 & 5 & 1 & 6 & 3 & 4 \\ v_2 & : & 9 & 5 & 6 & 4 & 8 & 3 & 2 & 7 & 1 \end{array}$$

$$\Rightarrow \begin{array}{lll|lll|lll} 7 & 9 & * & 2 & 5 & 1 & 6 & * & * \\ 9 & * & 4 & 2 & 8 & 3 & * & 7 & * \end{array}$$

$$\Rightarrow \begin{array}{lll|lll|lll} 7 & 9 & 2 & * & * & * & 5 & 1 & 6 \\ 9 & 6 & 4 & * & * & * & 8 & 3 & 7 \end{array}$$

$$\Rightarrow \begin{array}{llll|lll|lll} o_1 & 7 & 9 & 2 & 4 & 8 & 3 & 5 & 1 & 6 \\ o_2 & 9 & 6 & 4 & 2 & 5 & 1 & 8 & 3 & 7 \end{array}$$

- **Mutation Operator**
  After the new generation has been determined, the chromosomes are subjected to a low rate mutation process. For this example applies two mutation operators to introduce genetic diversity into the evolving population of permutation. The first operator is a simple two point mutation, which randomly selects two elements in the chromosome and swap them (1 10 **8** 4 5 6 7 9 **3** 2) becomes (1 10 **3** 4 5 6 7 9 **8** 2). The second operator is a shuffle mutation, which shunts the permutations forward by a random number of places; thus (**1 10 3 4 5** 6 7 9 8 2) shuffled forward six places becomes (6 7 9 3 2 **1 10 8 4 5**).

2. **Genetic Parameters**
   For MSP, from our experience, the following parameters are preferred to be used: population size (pop_size=20), probability of crossover (Pc=0.7), probability of mutation Pm =0.1 and some hundreds number of generations.

5.2 Use of PSO in MSP

For MSP, from our experience, the following parameters are preferred to be used: Number of Particles (N_Par=20,30), Maximum velocity (v_{max}=Number of Jobs (n)), Minimum velocity (v_{min}=1), Inertia Weight ($w\in[0.4,0.9]$). First acceleration parameter ($c_1\in[0.5,2]$), Second acceleration parameter ($c_2=c_1$), Diversity of the population Maintenance (random $r_1,r_2\in[0,1]$) and some hundreds number of generations.

6. Experimental Results of BAB, GA & PSO Implementation for (P_1)

For the problem (P_1), a simulation has been constructed in order to apply the BAB, GA & PSO.

Table (1) shows the CPU time results of applying BAB method in order to get a set of efficient solutions, on samples of different jobs with 10 experiments for each. The results of CPU time compared with results obtained from complete enumeration (CE) method, which generate all solutions for $n\leq 10$.

Table (1) the CPU time results of applying BAB and CE on (P_1) for n=7,..,15.

Table (1) the CPU time results of applying BAB and CE on (P_1) for n=7,..,15.

n	Experiments times/sec												CE
	1	2	3	4	5	6	7	8	9	10	Av	Av*	Av
7	0.01	0.0	0.08	0.02	0.01	0.06	0.01	0.05	0.05	0.04	0.03	0.03	0.01
8	0.25	0.01	0.40	0.14	0.06	0.09	0.03	0.01	0.05	0.09	0.11	0.11	0.63
9	0.07	0.06	0.03	0.01	0.03	0.03	0.18	0.02	0.39	0.2	0.11	0.11	6.32
10	0.12	0.08	**2.52**	0.09	0.10	1.60	0.04	0.30	0.57	0.40	0.58	0.37	63.1
11	0.31	**2.26**	1.12	0.18	0.08	0.70	0.06	0.07	0.34	0.03	0.52	0.32	---
12	0.14	1.43	3.00	0.25	0.22	0.70	2.27	0.19	1.26	**8.71**	1.82	1.05	---
13	**12.48**	0.40	0.26	0.35	6.76	0.49	0.36	3.67	0.20	1.58	2.65	1.50	---
14	1.838	40.3	4.11	2.01	139.6	0.73	8.97	1.61	0.07	**369.1**	56.7	22.1	---
15	6.18	1.31	1711	3.30	19.1	**8962**	1025	2.72	385.6	3.95	1212	351	---

Note:
1. The shaded cells are representing the most extreme time point.
2. Av.: denotes the average time for CE, where Av*.: is the average without the extreme time point for BAB.
3. No average time for CE method when n>10.

Table (2) shows the results of applying BAB on different examples of (P_3).

Table (2) the results of applying BAB on different examples of (P_3).

n	(efficient solutions of P_1)=($\Sigma C_i+\Sigma T_i$)=optimal Solutions of P_3	Lev.	Time/sec	
			Lev.	Tot.
20	(1075+742)=1817*,(1079+735)=1814,(1085+727)=1812,(1081+730)=1811, (1083+725)=1808,(1083+727)=1810,(1079+730)=1809,(1083+725)=1808, (1080+727)=1807,**(1081+725)=1806**	7	0.16	3.48
40	(3106+2381)=5487*,(3106+2379)=5485,(3111+2373)=5484, (3112+2371)=5483, **(3107+2375)=5482**	13	0.18	55.9
60	(7163+6142)=13305*,(7165+6139)=13304,(7164+6139)=13303, (7164+6138)=13302,(7166+6135)=13301,(7166+6134)=13300, (7169+6129)=13298,(7165+6131)=13296,(7172+6123)=13295, (7167+6127)=13294,(7166+6126)=13292,(7168+6122)=13290, (7167+6121)=13288,(7169+6117)=13286,(7168+6116)=13284, (7169+6114)=13283,(7170+6112)=13282,**(7169+6112)=13281**	13	33.9	---
80	(12404+10915)=23319*,(12409+10899)=23308,(12408+10899)=23307, (12414+10892)=23306,(12414+10888)=23302,(12412+10888)=23300, (12419+10880)=23299,(12416+10881)=23297,(12416+10880)=23296, (12420+10875)=23295,(12420+10874)=23294,(12420+10873)=23293, (12424+10868)=23292,(12423+10866)=23289,(12423+10865)=23288, (12425+10859)=23284,(12427+10853)=23280,(12429+10847)=23276, (12429+10844)=23273,(12429+10840)=23269,**(12429+10839)=23268**	18	1200	---

Notes:
1. The symbol * assigns the result obtained from applying SPT to (P_3) problem.
2. Lev. means the level of BAB with last optimal solution.
3. Tot. means the total time for applying BAB.

4. The symbol --- in time filed means the time which more than 30 minutes.
5. The bold value in solutions filed represents the optimal value of (P_3) problem.
6. Its very important result can obtained from finding the optimal solutions of (P_3) problem is that we can find the real (not approximate) efficient solutions of (P_1) problem for n=30,40,… with sizes can't obtained by applying modified BAB or any local search methods.

When using the parameters mentioned above, the best (near) efficient solutions of (P_1), time and number of iterations for best (near) efficient solutions of (P_1) results are showed, in table (3) and table (4) which are obtained when applying GA & PSO methods respectively, for number of jobs n=3,…,10, with number of generations, for 5 experiments for each number of jobs, using the following abbreviations:

1. **n**: **N**umber of jobs.
2. **Values of problem** (P_1):
   - **Ex**: **Ex**periment number.
   - **EffV**: **Eff**icient **V**alue(s) of (P_1) of each experiment using CE.
   - **BV**: **B**est **V**alue(s) of (P_1) of each experiment.
   - **MBV**: **M**inimum Best **V**alue.
   - **ABV**: **A**verage of **B**est **V**alues(s) of (P_1) for all experiments.
   - **ASPT**: **A**verage value of the **SPT** schedules for (P_1) for all experiments.
3. **AAE** : **A**verage of **A**bsolute **E**rror.

$$AAE = \frac{|EffV - ABV|}{EffV}$$

4. **Values of Time**:
   - **CT**: **C**omplete **T**ime for finishing each experiment.
   - **MCT**: **M**inimum **C**omplete **T**ime.
   - **BT**: **B**est **T**ime to obtain best value(s) of (P_1) of each experiment.
   - **MBT**: **M**inimum **B**est **T**ime.
   - **ABT**: **A**verage of **B**est **T**imes of (P_1) of all experiments.
5. **NI**: **N**umber of **I**terations of best value(s) of (P_1) of experiment.
6. **Number of Iterations**
   - **MNI**: **M**inimum **N**umber of **I**terations.
   - **ANI**: **A**verage **N**umber of **I**terations.
7. **Number of Efficient Solutions**:
   - **LES**: number of **L**ocal **E**fficient **S**olutions.
   - **RES**: number of **R**eal **E**fficient **S**olutions.
   - **ALES**: **A**verage of number of **L**ocal **E**fficient **S**olutions.

Table (3) Applying GA method on (P_1) for n=3,..,10.

n	**Ex**	**Values of (P_1)**			**Time/sec**			**NI**
		RES	**Best Value**	**ABV**	**CT**	**BT**	**ABT**	
3	1	(14,2)	(14,2)	(31,3)	0	0	0	1
	2	(35,4)	(35,4)		0	0		1
	3	(42,8),(46,6)	(42,8),(46,6)		0	0		1,2
	4	(33,0)	(33,0)		0	0		1
	5	(30,0)	(30,0)		0	0		1
4	1	(22,4)	(22,4)	(44,9)	0	0	0	1
	2	(62,15),(65,11)	(62,15),(65,11)		1	0		3,1
	3	(54,12)	(54,12)		0	0		6
	4	(44,5),(45,3),(51,2)	(44,5),(45,3),(51,2)		0	0		2,5,3
	5	(40,9),(46,7)	(40,9),(46,7)		0	0		1,22
5	1	(33,8),(39,5)	(33,8),(39,5)	(47,8)	0	0	0	1,2
	2	(47,7),(48,5),(49,2)	(47,7),(48,6),(48,5)		1	0		7,11,49
	3	(40,5),(43,1)	(40,5),(43,1)		0	0		1,30

	4	(48,6),(52,3),(56,1)	(48,6),(52,3)		0	0		2,13
	5	(68,15)	(68,15)		0	0		1
	1	(55,22),(59,19)	(55,22),(59,19)		1	0		2,3
	2	(70,15),(71,12),(72,11),(75,10),(76,9)	(71,12),(72,11),(75,10),(76,9)		0	0		2,30,39,54
6	3	(117,43)	(117,43)	(92,38)	1	0	0	3
	4	(104,51),(110,49)	(104,51),(110,49)		0	0		5,40
	5	(115,62),(116,58),(120,57),(121,56),(122,55)	(115,62),(116,58),(120,57),(121,56)		1	0		15,19,24,47
	1	(75,31),(80,29),(82,28),(83,26)	(75,31),(80,30),(82,28),(83,26)		2	0,1		3,57,131,141
	2	(76,23),(80,21),(84,19)	(76,23),(80,21),(84,19)		1	0,1		140,232,,257
7	3	(143,59),(147,58)	(143,59)	(101,41)	1	0	0	137
	4	(63,21),(64,20)	(64,20)		1	0		127
	5	(148,70),(150,68),(152,65)(155,63)	(148,70),(150,68),(152,65),(155,63)		1	0,1		2,172,173,191
	1	(89,44),(90,41), (93,40)	(89,44),(90,41),(93,40)		2	1,2,2		269,913,270
	2	(179,95),(181,94),(198,92),(201,91)	(179,95),(181,94),(199,93),(203,92)		1	1		158,42
8	3	(136,74),(137,73),(138,72),(140,70),(141,69),(142,68),(146,67),(153,66)	(136,74),(138,72)(140,70),(144,69),(146,68),(153,66)	(143,74)	2	1,2	1	11,132,801,480,747,393
	4	(194,114),(195,110),(196,108)	(194,114),(195,110),(196,111)		1	1		426,114,623
	5	(118,39)	(118,39)		1	1		208
	1	(115,45),(117,43),(135,41)	(115,45),(118,46)		4	3,2		1703,1462
	2	(246,123),(249,120),(254,118)	(246,123),(251,122),(256,121)		3	0,3		30,35,1145
9	3	(103,37),(107,34),(115,33)	(103,37),(107,34)	(180,92)	3	0,1	1	1338,159
	4	(243,149),(244,146),(245,144)	(243,149),(244,147)		3	1,2		556,1288
	5	(191,106),(192,102),(193,101)	(193,103)		2	1		1597
	1	(148,72),(149,69),(150,67),(153,66)	(150,67)		7	5		3124
	2	(147,55),(148,53),(159,52)	(148,59),(149,58)		7	3,4		2143,2706
10	3	(319,228)	(319,228)	(189,106)	7	3	4	1738
	4	(121,59),(122,57),(123,55),(124,54),(125,53),(127,52),(129,51),(131,50),(135,49)	(122,57),(123,55),(128,53),(135,50)		8	2,1,6,8		1405,325,4192,8520
	5	(207,116),(208,115),(211,114)	(207,117),(208,116)		7	7,5		5878,4415

Table (4) Applying PSO method on (P_1) for n=3,..,10.

n	**Ex**	**Values of (P_1)**			**Time/sec**			**NI**
		RES	**Best Value**	**ABV**	**CT**	**BT**	**ABT**	
	1	(14,2)	(14,2)		0	0		1
	2	(35,4)	(35,4)		0	0		1
3	3	(42,8),(46,6)	(42,8),(46,6)	(31,3)	1	0	0	1,2
	4	(33,0)	(33,0)		0	0		1
	5	(30,0)	(30,0)		0	0		1
	1	(22,4)	(22,4)		0	0		1
4	2	(62,15),(65,11)	(62,15),(65,11)	(44,9)	0	0	0	3,1
	3	(54,12)	(54,12)		0	0		1

	4	(44,5),(45,3),(51,2)	(44,5),(45,3),(51,2)		1	0		6,1,19
	5	(40,9),(46,7)	(40,9),(46,7)		0	0		2,15
5	1	(33,8),(39,5)	(33,8),(39,5)	(47,8)	0	0	0	37,3
	2	(47,7),(48,5),(49,2)	(47,7),(48,6),(49,2)		1	0		8,22,4
	3	(40,5),(43,1)	(40,5),(43,1)		0	0		13,98,5
	4	(48,6),(52,3),(56,1)	(48,6),(56,2)		0	0		12,89
	5	(68,15)	(68,15)		0	0		17
6	1	(55,22),(59,19)	(55,22),(63,20)	(93,38)	0	0	0	39,1
	2	(70,15),(71,12),(72,11), (75,10),(76,9)	(70,15),(71,12), (72,11),(76,9)		0	0		28,2, 20,42
	3	(117,43)	(117,43)		0	0		34
	4	(104,51),(110,49)	(104,51),(110,49)		0	0		79,73
	5	(115,62),(116,58),(120,57), (121,56),(122,55)	(117,58),(120,57), (121,56),(122,55)		0	0		6,73, 60,44
7	1	(75,31),(80,29),(82,28), (83,26)	(75,31),(82,28)	(101,41)	1	0,1	0	8,250
	2	(76,23),(80,21),(84,19)	(76,23),(80,21)		1	0,1		82,3
	3	(143,59),(147,58)	(143,59)		1	0		119
	4	(63,21),(64,20)	(64,20)		1	0		127
	5	(148,70),(150,68),(152,65) (155,63)	(148,70),(151,69), (152,65),(155,63)		1	0,1		258,249, 1,177
8	1	(89,44),(90,41),(93,40)	(90,41)	(144,72)	1	0	0	488
	2	(179,95),(181,94),(198,92), (201,91)	(179,95)		1	1		977
	3	(136,74),(137,73),(138,72), (140,70),(141,69),(142,68), (146,67),(153,66)	(136,74),(138,72) (141,69),(142,68)		1	0		104,72, 949,288
	4	(194,114),(195,110),(196,108)	(194,114),(197,108)		1	0		288,12
	5	(118,39)	(118,39)		1	0		546
9	1	(115,45),(117,43),(135,41)	(115,45),(134,43)	(181,92)	2	1	0	1333,107
	2	(246,123),(249,120),(254,118)	(246,123),(254,122)		2	0,1		222,946
	3	(103,37),(107,34),(115,33)	(103,37),(107,34), (117,33)		2	0,1		77,356, 834
	4	(243,149),(244,146),(245,144)	(245,151),(247,146)		2	1		1880,141
	5	(191,106),(192,102),(193,101)	(194,102)		1	0		829
10	1	(148,72),(149,69),(150,67), (153,66)	(151,77),(153,71), (154,67)	(190,107)	6	1,2	4	143,1693, 976
	2	(147,55),(148,53),(159,52)	(147,55)		5	1		747
	3	(319,228)	(321,229)		4	1		1563
	4	(121,59),(122,57),(123,55), (124,54),(125,53),(127,52), (129,51),(131,50),(135,49)	(122,57),(125,53), (133,51)		4	2,0, 3		3190,2, 4988
	5	(207,116),(108,115),(211,114)	(211,119)		4	0		364

Notes:

1. In tables (3,4), for n=3,..,6, 100 iterations are used, for 7, 500 iterations, for 8, 1000 iterations, for 9, 2500 iterations while for 10, 6000 iterations are being used.
2. It's important to note that the set of efficient solutions (POP) of (P_1) for each experiment obtained by using complete enumeration. The complete enumeration, of course, difficult to be applied for jobs more than 10 jobs. For this reason the results of complete enumeration are not mentioned in the tables (6,7) which are included more than 10 jobs.

In table (5) a comparison has been made between the results of applying GA (G) and PSO (P) obtained from tables (3,4) respectively, for values of (P_1), time and number of iterations for chosen n=4,7,10.

Table (5) Comparison results between GA & PSO methods on (P_1) for n=4,7,10.

n	**Ex**	**Values of (P_1)**					**ALES**		**Time/sec**						**NI**	
		EffV	**BV**		**ABV**		**LES/RES**		**CT**		**BT**		**ABT**			
			G	**P**	**G**	**P**	**G**	**P**	**G**	**P**	**G**	**P**	**G**	**P**	**G**	**P**
4	1	(22,4)	(22,4)	(22,4)					0	0	0	0			1	1
	2	(62,15) (65,11)	(62,15) (65,11)	(62,15) (65,11)			1.2	1.2	1	0	0	0			3 1	3 1
	3	(54,12)	(54,12)	(54,12)					0	0	0	0			6	1
	4	(44,5) (45,3) (51,2)	(44,5) (45,3) (51,2)	(44,5) (45,3) (51,2)	(44,9)	(44,9)	1	1	0	1	0	0	0	0	2 5 3	6 1 19
	5	(40,9) (46,7)	(40,9) (46,7)	(40,9) (46,7)					0	0	0	0			1 22	2 15
7	1	(75,31) (80,29) (82,28) (83,26)	(75,31) (80,30) (82,28) (83,26)	(75,31) (82,28)					2	1	0 1	0 1			3 57 131 141	8 250
	2	(76,23) (80,21) (84,19)	(76,23) (80,21) (84,19)	(76,23) (80,21)			2.6	2	1	1	0 1	0 1			140 232 257	82 3
	3	(143,59) (147,58)	(143,59)	(143,59)	(101,41)	(101,41)			1	1	0	0	0	0	137	119
	4	(63,21) (64,20)	(64,20)	(64,20)					1	1	0	0			127	127
	5	(148,70) (150,68) (152,65) (155,63)	(148,70) (150,68) (152,65) (155,63)	(148,70) (151,69) (152,65) (155,63)			0.9	0.7	1	1	0 1	0 1			2 172 173 191	258 249 1 177
10	1	(148,72) (149,69) (150,67) (153,66)	(150,67)	(151,77) (153,71) (154,67)					7	6	5	0			3124	143 1693 976
	2	(147,55) (148,53) (159,52)	(148,59) (149,58)	(147,55)			2.2	1	7	5	3 4	1 2			2143 2706	747
	3	(319,228)	(319,228)	(321,229)					7	4	3	1			1738	1563
	4	(121,59) (122,57) (123,55), (124,54) (125,53) (127,52) (129,51) (131,50) (135,49)	(122,57) (123,55) (128,53) (135,50)	(122,57) (125,53) (133,51)	(189,106)	(190,107)	0.6	0.4	8	4	2 1 6 8	1	4	0	1405 325 4192 8520	3190 2 4988
	5	(207,116) (108,115) (211,114)	(207,117) (208,116)	(211,119)					7	6	7 5	2 0			5878 4415	364

The average values, time and number of iterations for best values of problem (P_1) results are showed, in table (6) and table (7) which are obtained when applying GA and PSO methods respectively, from chosen number of jobs n=20(10)100, 100(100)1000 and 2000, for 10 experiments for each number of jobs.

Table (6) Applying GA method on (P_1) for chosen n=20,..,2000.

n	**Values of** (P_1)				**Time/sec**			**NI**	
	ASPT	**MBV**	**ABV**	**AAE**	**MCT**	**MBT**	**ABT**	**MNI**	**ANI**
20	(784,579)	(669,435)	(859,643)	(0.095,0.110)	1	0	1	419	1156
30	(1863,1528)	(1910,1572)	(2147,1798)	(0.152,0.176)	2	0	1	216	1007
40	(3279,2826)	(3133,2673)	(3847,3390)	(0.173,0.200)	1	0	2	93	1189
50	(5367,4767)	(5591,5037)	(6480,5861)	(0.207,0.230)	1	0	1	3	662
60	(7398,6718)	(8174,7477)	(9210,8506)	(0.245,0.266)	2	0	1	2	706
70	(9654,8859)	(11189,10378)	(12270,11456)	(0.271,0.293)	1	0	1	1	747
80	(11380,10457)	(13505,12486)	(14948,13993)	(0.314,0.338)	2	0	1	3	626
90	(16127,15088)	(19213,18103)	(20614,19545)	(0.278,0.295)	2	0	1	1	402
100	(20013,18877)	(23825,22750)	(26050,24887)	(0.302,0.318)	2	0	1	1	754
200	(76815,74457)	(99691,97318)	(103178,100788)	(0.343,0.354)	6	0	1	2	497
300	(172822,169340)	(225938,222610)	(237666,234143)	(0.375,0.383)	8	0	1	1	302
400	(316213,311418)	(401726,396955)	(430451,425620)	(0.361,0.367)	13	0	3	1	290
500	(486835,480862)	(647428,641474)	(672989,666986)	(0.382,0.387)	17	0	3	2	237
600	(694731,687549)	(942467,935042)	(963142,955924)	(0.386,0.390)	27	0	4	1	294
700	(946289,937878)	(1287845,1279258)	(1311518,1303071)	(0.386,0.389)	35	0	0	1	7
800	(1226999,1217444)	(1655960,1646540)	(1710498,1700911)	(0.394,0.397)	46	0	5	1	300
900	(1577954,1567071)	(2140353,2129737)	(2188328,2177419)	(0.387,0.390)	57	0	0	1	6
1000	(1927674,1915686)	(2610974,2599115)	(2682704,2670689)	(0.392,0.394)	70	0	0	1	9
2000	(7727745,7703649)	(10711403,10687729)	(10862863,10838728)	(0.406,0.407)	889	0	1	1	4

Table (7) Applying PSO method on (P_1) for chosen n=20,..,2000.

n	**Value of** (P_1)				**Time/sec**			**NI**	
	ASPT	**MBV**	**ABV**	**AAE**	**MCT**	**MBT**	**ABT**	**MNI**	**ANI**
20	(784,579)	(651,412)	(848, 636)	(0.081,0.099)	1	0	0	30	471
30	(1863,1528)	(1834,1509)	(2070,1726)	(0.111,0.129)	1	0	1	500	1181
40	(3279,2826)	(3076,2617)	(3751,3285)	(0.144,0.162)	1	0	1	81	1095
50	(5367,4767)	(5224,4663)	(6250,5629)	(0.164,0.181)	1	0	1	9	921
60	(7398,6718)	(7730,7035)	(8730,8038)	(0.180,0.197)	1	0	1	166	952
70	(9654,8859)	(0789,9984)	(11761,10951)	(0.218,0.236)	1	0	1	193	761
80	(11380,10457)	(12692,11702)	(14294,13350)	(0.256,0.277)	2	0	1	275	1004
90	(16127,15088)	(18248,17226)	(19828,18754)	(0.229,0.243)	2	0	1	78	845
100	(20013,18877)	(22844,21662)	(24940,23779)	(0.246,0.260)	2	0	2	5	1092
200	(76815,74457)	(94373,92003)	(100017, 97631)	(0.302,0.311)	5	0	3	207	1257
300	(172822,169340)	(220151,216721)	(229363,225847)	(0.327,0.334)	9	1	6	158	1744
400	(316213,311418)	(397074,392312)	(421116,416287)	(0.332,0.337)	13	0	5	169	1160
500	(486835,480862)	(630853,624771)	(657546,651548)	(0.351,0.355)	16	1	5	190	902
600	(694731,687549)	(926224,918964)	(942695,935485)	(0.357,0.361)	21	1	11	139	1408
700	(946289,937878)	(1265690,1257332)	(1287138,1278696)	(0.360,0.363)	27	1	6	56	699
800	(1226999,1217444)	(1621930,1612515)	(1676956,1667375)	(0.367,0.370)	32	0	16	19	1455
900	(1577954,1567071)	(2123093,2111958)	(2151797,2140887)	(0.364,0.366)	37	2	9	217	756
1000	(1927674,1915686)	(2589187,2577336)	(2644285,2632265)	(0.372,0.374)	53	1	19	44	1008
2000	(7727745,7703649)	(10521335,10497652)	(10706914,10682784)	(0.386,0.387)	228	6	104	228	1988

Note: In tables (6) and (7), for n=(20,10,100), 2000 iterations are used, for n=(200,100,1000), 3000 iterations, while for 2000, 5000 iterations are being used.

In table (8) a comparison has been made between the results of applying GA (G) and PSO (P) obtained from tables (6,7) respectively, for values of (P_1), time and number of iterations from chosen n=20(10)100, 100(100)1000 and 2000.

Table (8) Comparison results between GA & PSO on (P_1) for chosen n.

n	Value of (P_1)				Time				NI	
	ABV		AAE		MCT		ABT		ANI	
	G	P	G	P	G	P	G	P	G	P
40	(3847,3390)	(3751,3285)	(0.173,0.200)	(0.144,0.162)	1	1	2	1	1189	1095
70	(12270,11456)	(11761,10951)	(0.271,0.293)	(0.218,0.236)	1	1	1	1	747	761
100	(26050,24887)	(24940,23779)	(0.302,0.318)	(0.246,0.260)	2	2	1	2	754	1092
400	(430451,425620)	(421116,416287)	(0.361,0.367)	(0.332,0.337)	13	13	3	5	290	1160
700	(1311518,1303071)	(1287138,1278696)	(0.386,0.389)	(0.360,0.363)	35	27	0	6	7	699
1000	(2682704,2670689)	(2644285,2632265)	(0.392,0.394)	(0.372,0.374)	70	53	0	19	9	1008
2000	(10862863,10838728)	(10706914,10682784)	(0.406,0.407)	(0.386,0.387)	889	228	1	104	4	1988

Figure (2) describes comparison chart which shows the relation between value of problem (P_1) and number of iterations when applying GA & PSO on (P_1) for n=10.

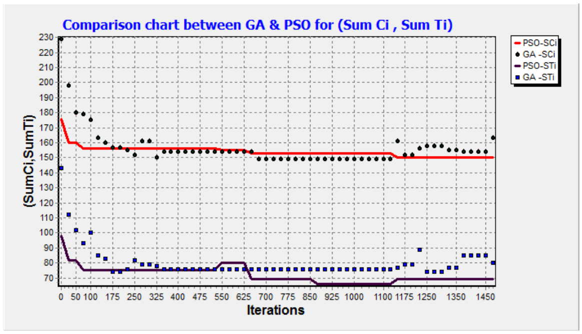

Figure (2) comparison chart of applying GA & PSO on (P_1), n=10, NI=1500.

Figure (3) describes comparison chart which shows the relation between values of (P_1) and number of iterations when applying GA & PSO for n=150.

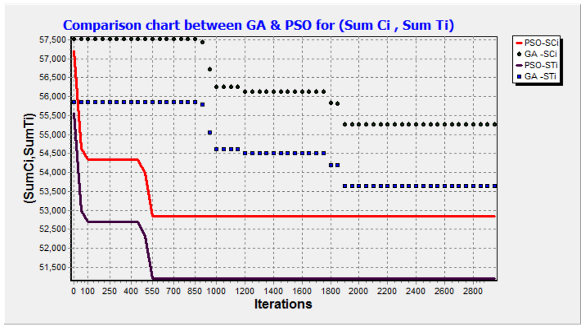

Figure (3) comparison chart of applying GA & PSO on (P_1), n=150, NI=3000.

Figure (4) describes comparison chart which is shows the relation between values of AAE of (P_1) and number of iterations when applying GA & PSO for n=180.

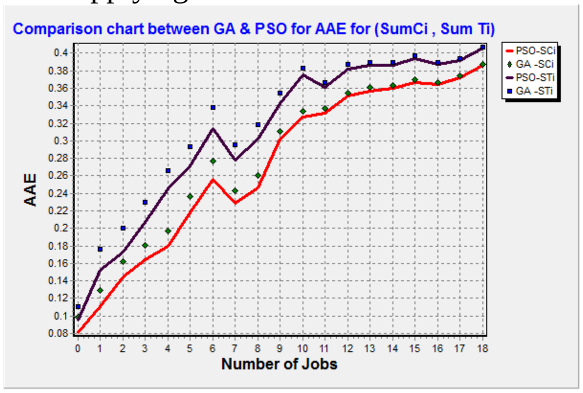

Figure (4) comparison chart of applying GA & PSO on (P_1) for AAE, n=180.

7. Analysis of Results

1. In this paper, a different number of jobs (n) are used, starting from n=4(1)10, n=20(10)100, n=200(100)1000 and n=2000, with number of iterations which is suitable to n to solve problem (P_1).
2. The parameters of testing the efficiency of local search method (GA and PSO) are calculated, these parameters represented by, the pair ($\Sigma C_i, \Sigma T_i$) of real efficient solutions (are calculated from complete search method for $n \leq 10$), the approximated local search best efficient solutions (BV) and their average, the number of best solutions (LES) and their average, the average of absolute error, the time which complete the single experiment, the time of the BV and their average and lastly, the iteration which found the corresponding BV.
3. From table (5), for chosen n,
   - Number of LES: GA is serves better than PSO.
   - The two algorithms are equals in accuracy of resultant efficient solutions.
   - The two algorithms are approximately equals in CT and ABT.
   - The results of the iterations of ABV are different and unstable for each algorithm.
4. Figure (2) (for n=10), describes an approximated result in the results of BV for GA & PSO.
5. From table (8), for chosen n (20,..,2000),
   - We calculate the efficiency of BV obtained from the two algorithms compared with efficient of SPT by using AAE, we can conclude that PSO is better.
   - PSO is better than GA in ACT.
   - The results of the iterations of ABV are different and unstable for each algorithm.
6. Figure (3) (for n=150), shows that PSO is better in BV.
7. Figure (4) (for n=180), illustrates the efficiency of PSO in giving better accuracy than GA, but the results are closed and the AAE increased (accuracy are decreased) when the number of jobs are increased.

8. BAB, GA & PSO System Requirements for Solving problem (P_1)

The BAB, GA & PSO methods were tested by programming them using version 10.0 of Delphi Language and MATLAB, and running on Processor Intel(R) Core(TM) i3 CPU, 2.53 GHz, Core(s), with Ram 1.21 GB computer.

9. Conclusions

1. From the results obtained for applying the two local search to find an efficient solutions for problem (P_1), we can conclude that GA is better in n=3,..,10 (low number of jobs), while PSO is better in n=20,..,2000 (higher number of jobs).
2. The size of solution space (n!) can be decreasing when applying the precedence rules on each scheduling σ before join it to the population, since some of the efficient solutions are satisfy the precedence rules.
3. To improve the performance of GA and PSO, we suggest making a hybrid between the two algorithms from one side, or between them and another local search algorithm e.g. simulated annealing and Bee algorithm, from other side.

References

(1) Chang P, and Su L., (2001). Scheduling n Jobs on One Machine to Minimize the Maximum Lateness with Minimum Number of Tardy Jobs, Computer & Industrial Engineering, 40, 49-60.

(2) Chen T, Qi X. and Tu F., (1997). Single Machine Scheduling to Minimize Weight Earliness Subject to Maximum Tardiness, Computer of Operation Research, 24, 147-152.

(3) Van Wassenhove L. N., and Gelders F., (1980). Solving a Bi-Criteria Scheduling Problem, European Journal of Operation Research 4/1, 42-48.

(4) Hoogeveen J. A., (1996). Single Machine Scheduling to Minimize a Function of Two or Three Maximum Cost Criteria, Journal of Algorithms, 21, 415-433.

(5) Al-Assaf, S. S., (2007). Solving multiple objective scheduling problems, M. Sc. Thesis, University of Al-Mustansiriyah, College of Science, Dept. of Mathematics.

(6) Findi F. S., (2012). Minimization of Multicriteria Scheduling in One Machine, M. Sc. Thesis, University of Baghdad, College of Education, Ibn Al-Haitham, Dept. of Mathematics.

(7) Shi Y., (2004). Particle Swarm Optimization, Electronic Data Systems, Inc. Kokomo, IN 46902, USA Feature Article, IEEE Neural Networks Society.

(8) Sabah M. S., (2004). A Comparative Study between Traditional Genetic Algorithms and Breeder Genetic Algorithms, M. Sc. Thesis, AL-Nahrain University.

(9) Rinooy Kan, A. H. G., (1976). Machine Sequencing Problem: Classification, Complexity, and Computation, Nijhoff, the Hague.

(10) Smith W. E., (1956). Various Optimizer for Single-Stage Production, Naval Res. Logist. Quart.3, 59-66.

(11) Jouni L., (2000). Multi-Objective Nonlinear Pareto-Optimization, Lappeenranta University of Technology.

(12) Hoogeveen J. A., (2005). Invited Review Multicriteria Scheduling, European Journal of Operation Research 167, 592-623.

(13) Hoogeveen H., (2005). Multicriteia Scheduling, Department of Computer, Utrecht University, P.O. Box 80089, Utrecht 3508TB, Netherlands.

(14) Pindeo, M., (2002). Scheduling: Theory; Algorithms and Systems, Printice-Hill, Inc., Englewood Diffs, New Jersey 2nd Edition.

(15) Evans, G. W., (1984). An Overview of Techniques for solving Multi-Objective Mathematical Programs, Management Science, Vol. 30, pp.1268-1282.

(16) Dileepan P. and Sent T., Bicriterion, (1988). Static Scheduling Research for a Single Machine, OMEGA; 16(1):53-59.

(17) Fry T. D., Armstrong R. D. and Lewis H., (1989). A Framework for Single Machine Multiple Objective Sequencing Research, OMEGA; 17 (6): pp.595-607.

(18) Hoogeveen J. A., (1992). Single Bi-Criteria Scheduling, Ph. D. Dissertation, Center for Mathematics and Computer Science, Amsterdam, Netherlands.

(19) Lee C. V., and Vairaktarakis G. L., (1993). Complexity of Single Machine Hierarchical Scheduling: A Survey, Report No. 95-10, Department of Industrial and Systems Engineering, University of Florida, Gainesville FL, USA.

(20) Nagar A., Haddock J. and Heragu S., (1995). Multiple and Bi-Criteria Scheduling a Literature Survey, Eur. J. OPI. Res. 1:88-104.
(21) Eiben A. and Smith J., (2007). Introduction to Evolutionary Computing. Springer, 2nd edition.
(22) Auger A. and B. Doerr, (2011). Theory of Randomized Search Heuristics: Foundations and Recent Developments. World Scientific.
(23) Ali F. H., (2007). Cryptanalysis of the Stream Cipher Systems Using the Genetic Algorithm, Information Technology & National Security Conference, Al-Riyadh-KSA, 1-4/Dec./2007.
(24) Kennedy J. and Eberhart R. C. (1995). Particle Swarm Optimization, Proceedings of IEEE International Conference on NN, Piscataway, pp. 1942-1948.
(25) Ribeiro P. F. and Kyle W. S., (2003). A Hybrid Particle Swarm and Neural Network Approach for Reactive Power Control, Member. http://engr.calvin.edu/…/courses/engir302/Reactivepower-PSO-wks.pdf.
(26) Woodruft D. L., (1998). Advanced in Computational and Stochastic Optimization. Logic Programming, and Heuristic search.

خوارزميات تقليل وقت الاتمام الكلي والتأخير الكلي لجدولة ماكنة واحدة

د.طارق صالح عبد الرزاق وفائز حسن علي

قسم الرياضيات، كلية العلوم، الجامعة المستنصرية، بغداد، العراق.

<u>الخلاصة</u>

في هذا البحث سيتم مناقشة مسالة جدولة n من الاعمال لها اوقات تنفيذ والوقت المثالي لانجاز النتاج لماكنة واحدة. الهدف هو ايجاد جدولة تقلل قيمة دالة مجموع وقت الاتمام ومجموع وقت التأخير (لتقليل دالة متعددة الاهداف ($\Sigma C_i,\Sigma T_i$)). في هذا البحث نقترح طريقتين لحل مسالة التقليل ألآني لايجاد مجموعة كل الحلول الكفوءة (حلول باريتو المثالية). ان ايجاد مجموعة الحلول الكفوءة ليس بالامر الهين، لذلك، من الافضل ايجاد قيم تقريبية لمجموعة الحلول وفي اوقات معقولة. لذلك تم استخدام طريقة التقيد والتفرع (BAB) وطرق البحث المحلية.

تم تطبيق طريقة امثلية السرب الجزيئي (PSO)، طريقة بحث محلية جديدة، على مسائل مولدة عشوائياً لحل مسائل مكائن الجدولة متعددة الاهداف. ولان مسألتنا هي من المسائل المعقدة، فاننا نقترح استخدام طرق تقريبية جديدة مثل (PSO) و(GA) لايجاد حلول تقريبية خصوصا عندما يتجاوز عدد الاعمال امكانية بعض الطرق الحل التام مثل حل التام مثل طريقة العد التام وطريقة (BAB). تم اجراء دراسة مقارنة بين طريقة التقيد والتفرع وطريقة امثلية السرب الجزيئي والخوارزمية الجينية لبيان اي منها الافضل عند التطبيق.