Dynamic Programming Algorithms for Solving Single Machine Scheduling Problems with a Fixed Complexity Time

Kawa A. Al-Qazilchy
College of Sciences
University of Salahaddin

Received on: 19/08/2007 **Accepted on: 17/12/2007**

ABSTRACT

We solve in this paper some machine scheduling problems with certain conditions; the solution had been made by using dynamic programming procedures with a fixed time for all the algorithms. Some of the algorithms described in the paper represent a modification to some mathematical programming algorithms and some of them are new or a generalization to others. The optimal solution of all these algorithms will be done with a fixed complexity time, which may lead us to solve them easier and faster.

Key words: scheduling, single machine, set-up times, dynamic programming, fixed complexity.

خوارزميات البرمجة الديناميكية لحل مسائل جدولة الماكنة الواحدة بوقت حسابي ثابت

كاوه الكزلجي
كلية العلوم، جامعة صلاح الدين

تاريخ الاستلام: 2007/8/19 **تاريخ القبول: 2007/12/17**

الملخص

تناولنا في هذا البحث حل بعض مسائل جدولة الماكنه بوجود شروط معينه, تم الحل باستعمال خوارزميات برمجه ديناميكيه و بوقت ثابت لكل الخوارزميات. الخوارزميات الموصوفة في هذا البحث تمثل خوارزميات جديدة أو تطويرية. الحل الأمثل لجميع الخوارزميات سوف يتم بوقت حسابي ثابت و الذي سوف يقودنا الى حلها حلاً أسهل و أسرع.

الكلمات المفتاحية: الجدولة، الماكنة الواحدة، وقت الإعداد، البرمجة الديناميكية، الحسابي الثابت.

1. Introduction

In the recent years, there has been a large importance in scheduling problems with set up times. The set up times occur when we have different families of jobs, as an example of families and set ups; this may happen in an industrial unit of coloring or painting plastic or wood products. Products (jobs) with the same colors may represent a one class (family). Cleaning the machine from a particular color to receive another different product color will represent a set-up time so the products with the same color will represent a family, the families of jobs may possibly take place in other fields of the life. We derive in this paper some dynamic programming algorithms to solve single machine scheduling problems with set up time families with an order of complexity which is fixed for all our dynamic programming algorithms, in other words with the same complexity algorithm time which may help us to solve the problems more rapidly. Before describing the dynamic programming algorithms we will use the standard classification scheme for scheduling problems (Graham et al. [7]) which is $\alpha/\beta/\varphi$, where α indicates the machine environment, in this paper α will be equal to 1, which indicates that we will have single machine problems(α=1), β describes the job and family characteristics and the constraints on the jobs, and φ defines the objective function to be minimized. Before describing the algorithms, let {1, 2, …,n} denote the set of jobs to be processed, also for each family f (f=1, 2, ...,F), we will define the following parameters: $p_{i\,f}, d_{i\,f}$ *and* $w_{i\,f}$ which represent the processing time, the due date and the weight of the ith job of the fth family, respectively. Also for each family f we will assign a positive specific amount, which is called the set-up time s_f.

For each job i we will define the following variables:

C_i	: is the completion time of job i,
$L_i = C_i - d_i$	: is the lateness of job i,
$T_i = \max\{L_i, 0\}$	: is the tardiness of job i.

Potts and Kovalyov [10] gave a preview of solving some machine scheduling problems for single machine and more than one machine, by using dynamic programming algorithms, the rest of this paper will be organized as follows: First we will give in section 2, a general description of our algorithm and then we will give the complexity time of the algorithm. In section 3, we will discuss our dynamic programming algorithms. This discussion may be made as follows:- We will discuss the dynamic algorithm for the sum of completion times problem $1/s_f/\sum C_i$ which is a generalization to that one which is described in [9], moreover in the same section, we will describe the sum of the weighted completion times problem $1/s_f/\sum w_i C_i$ and solve it by another dynamic programming algorithm which will represent a generalization to that which is described in [1], we will use a dynamic programming algorithm to solve the square completion times problem $1/s_f/\sum C_i^2$, in the same section, an algorithm of finding the optimal solution of the weighted summation of square completion times, with equal weights $1/s_f, w_{i\,f} = w_f/\sum w_i C_i^2$, and the weighted sum of square completion times, with equal processing times $1/s_f, p_{i\,f} = p_f/\sum w_i C_i^2$ will also be considered in this section. In the same section, we will use a dynamic programming algorithm to solve the sum of tardy jobs with equal due dates 1/ s_f, $d_{i\,f}$ = d_f / $\sum T_i$, and to solve the sum of tardy jobs with equal processing times 1/ s_f, $p_{i\,f}$ = p_f / $\sum T_i$, also in the same section, we will describe a dynamic programming algorithm to solve the weighted summation of tardy jobs with equal weighted due dates 1/ s_f, $w_{i\,f}\, d_{i\,f} = w_f\, d_f$ / $\sum w_i T_i$ .Finally the same section, will describe an algorithm which finds the optimal solution of the maximum lateness 1/s_f/L_{max}. At last section 4 gives us some conclusions on the algorithms.

2. Algorithms Description and Complexity Time

The dynamic programming algorithms which have been discussed in this paper; compute values $g(n_1, n_2, ..., n_f, ..., n_F, f)$ for f = 1, 2, ..., F, these values represent the minimum total overall contribution to the cost of the objective function, when n_f job(s) from family f (f = 1, 2, …, F) is(are) sequenced, the last term f of the recursive equations g , indicates that the last sequenced job is from family f (f = 1, 2, …, F). For example the value g (2, 1, 1, 2) will indicate that we have three families where two jobs from family 1, one job from family 2 and one job from family 3 are sequenced, the last term of the value g (2, 1, 1, 2) is equal to two, which indicates that the last sequenced job is from family 2. Suppose that the current state is $(n_1, n_2, ..., n_f, ..., n_F, 1)$ where $n_1 \geq 1$, in that state we see f is equal to 1(f=1), which means that the last scheduled job is from the first family, this means that the n_1th job of the first family which can be denoted by (n_1, 1) is scheduled last. The previous state is either $(n_1 - 1, n_2, ..., n_f, ..., n_F, 1)$ or it is any one of the states $(n_1 - 1, n_2, ..., n_f, ..., n_F, h)$ where h is a family which is distinct from the first family, that is $h \in \{2, 3, ..., F\} (h \neq 1)$. If the previous state is $(n_1 - 1, n_2, ..., n_f, ..., n_F, 1)$, then no set up would be required before the job (n_1, 1), so its overall contribution to the objective function will become:

$$(W - W_1 - W_2 - ... - W_F)\ \text{cost}(p_{n_1 1}) \qquad ...(1)$$

where $W = \sum_{i=1}^{n} w_i$, $W_1 = \sum_{i=1}^{n_1} w_{i-1}$ and $W_h = \sum_{i=1}^{n_h+1} w_{i-1}$ for all $h \neq 1$. The cost here is one of our objective functions which are described in the previous section.

Some of the machine scheduling problems will have unit weights, in other words these jobs share the same weight (importance), and for this reason expression 1 will be converted to:

$$(n - n_1 - n_2 - ... - n_F + 1)\ \text{cost}(p_{n_1 1}) \qquad ...(2)$$

Now to find all the values of $g(n_1, n_2, ..., n_f, ..., n_F, f)$, we must apply the following recursions:

$$g(n_1, n_2, ..., n_f, ..., n_F, f) = \min\{g(n_1, n_2, ..., n_f - 1, ..., n_F, f) + W_f \cos t(p_{n_f f}), \min_{h \neq f}\{g(n_1, n_2, ..., n_f - 1, ..., n_F, h) + W_f \cos t(s_f + p_{n_f f})\}\} \quad ...(3)$$

where $W_f = W - \sum_{i=1}^{n_1} w_{i-1} - \sum_{i=1}^{n_2} w_{i-1} - ... - \sum_{i=1}^{n_f} w_i - ... - \sum_{i=1}^{n_F} w_{i-1}$, w_0=0 for f = 1, 2, …, F. Initial values are $g(1, 0,..,1) = W \cos t(s_1 + p_{1,1})$, $g(0, 1,.., 2) = W \cos t(s_2 + p_{1,2})$ and … $g(0, 0,...,1, F) = W \cos t(s_F + p_{1,F})$. The non existence of some values of the recursive equations g will lead us to let them be equal to infinity that is all other initial values are set to infinity, observe that if the recursive equations are of the maximum type then we must let some of the values equal to minus infinity, we will explain the non existence of these values in the next section.

Our aim is to determine:

$$\min\{g(q_1, q_2, ..., q_F, 1),\ g(q_1, q_2, ..., q_F, 2), ...,\ g(q_1, q_2, ..., q_F, F)\}$$

Where q_f is the total number of the jobs of family f (f = 1, 2, …, F).

The time complexity of the dynamic programming algorithms will depend on the number of the states of the recursive equations which described in (3), the desired time can be described as follows, for each family f, we have $n_f \in \{0, 1, ..., q_f\}$, so for all the F families, we will have $(1+q_1)\ (1+q_2)\ …\ (1+q_F)$ states, we observe that for each state the corresponding recursive equation is solved in a constant time. Now since we have $q_1 < n_1$, $q_2 < n_2$, …, $q_F < n_F$, so we deduce that the algorithm will require $O(n_1.n_2.n_3. … .n_F)$ time, now since $n_f < n$ for all f then we deduce that the algorithm would be of $O(n^F)$ complexity time. All the algorithms described in this paper will share a fixed time of complexity which is n^F as mentioned above, so rather than solving the algorithms in different times it is better to solve them in fixed computation time, hence the fixed time will lead us to solve these algorithms faster and easier.

3. Dynamic Algorithms for the Objective Functions

This section will be devoted to describe the dynamic programming algorithms of our objective functions.

3.1. The sum of the completion and weighted completion times

The dynamic programming algorithm for the sum of completion times problem and the dynamic programming algorithm of the sum of weighted completion times problem are considered first. Before describing the recursive equations we must first rearrange the jobs of each family in non decreasing order of the processing time [8], the optimal solution of the 1/ s_f / $\sum C_i$ problem can be described as:

$$g(n_1,n_2,...,n_f,...,n_F,f)=\min\{g(n_1,n_2,...,n_f-1,...,n_F,f)+N(p_{n_f f}),\\ \min_{h\neq f}\{g(n_1,n_2,...,n_f-1,...,n_F,h)+N(s_f+p_{n_f f})\}\} \qquad ...(4)$$

where N = n_1 - n_2 - … - n_F + 1, initial values are $g(1,0,..,1)=N(s_1+p_{1,1})$, $g(0,1,..,2)=N(s_2+p_{1,2})$ and … $g(0,0,...,1,F)=N(s_F+p_{1,F})$, now since some of the cases will be impossible, for instance the case g(1, 0, …, 2) means that the last sequenced job is from the family 2 but we have no any sequenced job from this family so we set the value of this case to be infinity, that is the reason of setting all other initial values to infinity.

The recursive equations of Potts [9] were given to solve the sum of completion times with only two families and these recursions are:

$$g(n_1,n_2,1)=\min\{g(n_1-1,n_2,1)+N(p_{n_1 1}),\ g(n_1-1,n_2,2)+N(s_1+p_{n_1 1})\}$$
$$g(n_1,n_2,2)=\min\{g(n_1,n_2-1,2)+N(p_{n_2 2}),\ g(n_1,n_2-1,1)+N(s_2+p_{n_2 2})\}$$

We generalize the idea of Potts [9] to more than two families in other words to F families as mentioned in the recursive equations (4).

Before applying the recursive equations of the sum of the weighted completion times, the jobs of each family must be sequenced in non decreasing order of the ratios p_{if}/w_{if} [8]. The optimal sequence of the sum of weighted completion times 1/ s_f / $\sum w_i C_i$ can be determined by applying the following recursive equations:

$$g(n_1,n_2,...,n_f,...,n_F,f)=\min\{g(n_1,n_2,...,n_f-1,...,n_F,f)+W_f(p_{n_f f}),\\ \min_{h\neq f}\{g(n_1,n_2,...,n_f-1,...,n_F,h)+W_f(s_f+p_{n_f f})\}\} \qquad ...(5)$$

where $W_f=W-\sum_{i=1}^{n_1} w_{i-1}-\sum_{i=1}^{n_2} w_{i-1}-...-\sum_{i=1}^{n_f} w_i-...-\sum_{i=1}^{n_F} w_{i-1}$, w_0=0 for f = 1, 2, …, F. Initial values are $g(1,0,..,1)=W(s_1+p_{11})$, $g(0,1,..,2)=W(s_2+p_{12})$ and … $g(0,0,...,1,F)=W(s_F+p_{1F})$, as mentioned above all other initial values are set to infinity.

The recursive equations of Abdullah [1] were given to solve the sum of completion times with only two families and these recursions are:

$$g(n_1,n_2,1)=\min\{g(n_1-1,n_2,1)+W_f(p_{n_1 1}),\ g(n_1-1,n_2,2)+W_f(s_1+p_{n_1 1})\}$$
$$g(n_1,n_2,2)=\min\{g(n_1,n_2-1,2)+W_f(p_{n_2 2}),\ g(n_1,n_2-1,1)+W_f(s_2+p_{n_2 2})\}$$

We generalize the idea of Abdullah [1] to more than two families in other words to F families as mentioned in the recursive equations (4).

Crauwels et el. [4] and Abdul-Razaq and Abdullah [2] propose branch and bund algorithms for the problem 1/ $s_f / \sum w_i C_i$, also this problem has been solved by an O(n^F) algorithm by Ghosh [5], which is equivalent to our algorithm, Ahn and Hyun [3], propose a forward dynamic programming algorithm with job appending that also requires O(n^F) time.

3.2. The sum of the square completion and weighted square completion times

The optimal solution of the sum of the square completion times problem / $s_f / \sum C_i^{\,2}$, can be determined first by rearranging the jobs according to non decreasing order of the processing times p_{if}, see [11], and second by applying the following recursive equations:

$$g(n_1,n_2,...,n_f,...,n_F,f) = \min\{g(n_1,n_2,...,n_f-1,...,n_F,f)+N(p_{n_f f})^2, \min_{h\neq f}\{g(n_1,n_2,...,n_f-1,...,n_F,h)+N(s_f+p_{n_f f})^2\}\} \quad \ldots(6)$$

Where N = $n_1 - n_2 - \ldots - n_F + 1$, initial values are $g(1,0,..,1) = N(s_1+p_{1,1})^2$, $g(0,1,..,2) = N(s_2+p_{1,2})^2$ and $\ldots\ g(0,0,...,1,F) = N(s_F+p_{1,F})^2$, all other initial values are set to infinity.

We can get the optimal solution of the sum of the weighted square completion times problem with equal processing times 1/s_f, p_{if}=p_f / $\sum w_i C_i^{\,2}$, by rearranging the jobs of each family f according to non increasing order of w_{if}, and then by applying the following recursions:

$$g(n_1,n_2,...,n_f,...,n_F,f) = \min\{g(n_1,n_2,...,n_f-1,...,n_F,f)+W_f(p_{n_f f})^2, \min_{h\neq f}\{g(n_1,n_2,...,n_f-1,...,n_F,h)+W_f(s_f+p_{n_f f})^2\}\} \quad \ldots(7)$$

Where $W_f = W - \sum_{i=1}^{n_1} w_{i-1} - \sum_{i=1}^{n_2} w_{i-1} - \ldots - \sum_{i=1}^{n_f} w_i - \ldots - \sum_{i=1}^{n_F} w_{i-1}$, w_0=0 for f = 1, 2, …, F. Initial values are $g(1,0,..,1) = W(s_1+p_{1,1})^2$, $g(0,1,..,2) = W(s_2+p_{1,2})^2$ and … $g(0,0,...,1,F) = W(s_F+p_{1,F})^2$, all other initial values are set to infinity.

3.3. The sum of the tardy and weighted tardy jobs

Next in this section we will find the optimal solution of the sum of tardiness jobs with equal due dates 1/ s_f, $d_{i\,f} = d_f / \sum T_i$, we see that:

$$\begin{aligned} & T_i = \ Max\{C_i - d_i, 0\} && \textit{(definition of } T_i) \\ \Rightarrow\quad & \sum T_i = \sum Max\{C_i - d_i, 0\} && \textit{(by taking the sum of both sides)} \qquad \ldots(8) \\ \Rightarrow\quad & \sum T_i = \sum Max\{C_i - d, 0\} && \textit{(by assumption)} \end{aligned}$$

We notice from equation (8) that since d is a fixed due date time then we can rearrange the jobs of each family by non decreasing order of the processing times p_{if}, the rearrangement will lead us to the best solution, our aim here is to find the optimal solution, and this can be reached by applying the following recursive equations:

$$\begin{aligned} g(n_1,n_2,\ldots,n_f,\ldots,n_F,f) = \min\{ & g(n_1,n_2,\ldots,n_f-1,\ldots,n_F,f) + \\ & \max\{g_c(n_1,n_2,\ldots,n_f-1,\ldots,n_F,f) + p_{n_f f} - d_f, 0\} \\ & \min_{h\neq f}\{g(n_1,n_2,\ldots,n_f-1,\ldots,n_F,h) + \\ & \max\{g_c(n_1,n_2,\ldots,n_f-1,\ldots,n_F,h) + s_f + p_{n_f f} - d_f, 0\}\} \end{aligned} \qquad \ldots(9)$$

where $g_c(n_1,n_2,\ldots,n_f-1,\ldots,n_F,f)$ is the completion time of the n_f-1's job of family f (f=1, 2, …, F) which can be obtained from the previous state of the recursive equations that is from the minimum solution of the preceding state.

To explain how the recursive equations (9) work, let us take the following example: consider a problem with three families and six jobs with the following data:

i	1	2	3	4	5	6
p_i	4	2	2	4	3	5
d_i	5	5	7	7	8	8

Table 1

and the families are f_1={1, 2}, f_2={3, 4} and f_3={5, 6}, with set-up times s_1=2, s_2=3 and s_3=2 of the three families, respectively. Now since $p_2<p_1$, so the order of the jobs of family 1 becomes 2, 1, that is f_1 must become {2, 1}, while the order of the jobs of the families f_2 and f_3 will remain as they are. To explain the method more precisely let us compute the value of g(2, 1, 0, 1) which is:

$$\begin{aligned} g(2, 1, 0, 1) = \min\{ & g(1, 1, 0, 1) = 2 + \max\{ g_c(1, 1, 0, 1) = 9 + 4 - 5 , 0\}, \\ & g(1, 1, 0, 2) = 2 + \max\{ g_c(1, 1, 0, 2) = 9 + 2 + 4 - 5, 0\}, \\ & g(1, 1, 0, 3)=\infty + \max\{ g_c(1, 1, 0, 3)=\infty+2 + 4-5,0\}\}=10 \end{aligned}$$

where the values of g(1, 1, 0, 1), g(1, 1, 0, 2), g(1, 1, 0, 3), g_c(1, 1, 0, 1), g_c(1, 1, 0, 2) and g_c(1, 1, 0, 3) can be obtained from previous stages. The optimal solution can be found at the last stage of the recursive equations that is the goal here is to find min{g(2, 2, 2, 1), g(2, 2, 2, 2), g(2, 2, 2, 3)}, we notice that the optimal solution is 52, backtracking which illustrated in figure 1 shows that the optimal sequence is (2, 1, 3, 4, 5, 6).

$$\begin{tikzcd}
\boxed{g(1,0,0,1)=4} \\
\boxed{g(2,0,0,1)=3} \arrow[u] \\
\boxed{g(2,1,0,2)=9} \arrow[u] \\
\boxed{g(2,2,0,2)=19} \arrow[u] \\
\boxed{g(2,2,1,3)=33} \arrow[u] \\
\boxed{g(2,2,2,3)=52} \arrow[u]
\end{tikzcd}$$

Figure 1

The $1/\ s_f,\ p_{i\,f} = p_f\ /\sum T_i$ problem can be solved first by rearranging the jobs according to non decreasing order of $d_{i\,f}$ and second by applying the following recursive equations:

$$\begin{aligned}
g(n_1,n_2,\dots,n_f,\dots,n_F,f) = \min\{ & g(n_1,n_2,\dots,n_f-1,\dots,n_F,f)+ \\
& \max\{g_c(n_1,n_2,\dots,n_f-1,\dots,n_F,f)+p_f-d_{n_f f},0\} \\
& \min_{h\neq f}\{g(n_1,n_2,\dots,n_f-1,\dots,n_F,h)+ \\
& \max\{g_c(n_1,n_2,\dots,n_f-1,\dots,n_F,h)+s_f+p_f-d_{n_f f},0\}\}
\end{aligned} \quad \dots(10)$$

The sum of the weighted tardy jobs problem with equal weighted due dates $1/\ s_f,\ w_{i\,f}\, d_{i\,f} = w_f\, d_f\ /\ \sum w_i T_i$ will be considered now, here we have:

$$\begin{aligned}
& T_i = \quad Max\{C_i - d_i, 0\} && (definition\ of\ T_i) \\
\Rightarrow\ & w_i T_i = \quad w_i\, Max\{C_i - d_i, 0\} && (by\ multiplying\ both\ sides\ by\ w_i) \\
\Rightarrow\ & \sum w_i T_i = \sum w_i\, Max\{C_i - d_i, 0\} && (by\ taking\ the\ sum\ of\ both\ sides) \quad \ldots(11) \\
\Rightarrow\ & \sum w_i T_i = \sum Max\{w_i\, C_i - w_i\, d_i,\ 0\} && (\text{sin}\,ce\ w_i\ is\ a\ positive\ weight) \\
\Rightarrow\ & \sum w_i T_i = \sum Max\{w_i\, C_i - w\, d,\ 0\} && (by\ assumption)
\end{aligned}$$

Now from the last equation of (11) we notice that since wd is a fixed weighted due date time for all the families so rearranging the jobs of each family according to non decreasing order of p/w and then applying the recursive equations (12) will give us the optimal solution of the last problem.

$$\begin{aligned}
g(n_1,n_2,\ldots,n_f,\ldots,n_F,f) = \min\{ & g(n_1,n_2,\ldots,n_f-1,\ldots,n_F,f) + \\
& \max\{g_w(n_1,n_2,\ldots,n_f-1,\ldots,n_F,f) + p_{n_f f} - w_f d_f, 0\} \\
& \min_{h\neq f}\{g(n_1,n_2,\ldots,n_f-1,\ldots,n_F,h) + \qquad\qquad \ldots(12) \\
& \max\{g_w(n_1,n_2,\ldots,n_f-1,\ldots,n_F,h) + s_f + p_{n_f f} - w_f d_f, 0\}\}
\end{aligned}$$

To the best of our knowledge the algorithms which are described in this section are new ones.

3.4. The maximum lateness

The last problem which is considered in this paper is maximum lateness with set-up times that is the $1/s_f/L_{max}$ problem. This problem can be solved by applying the following steps: the first step is to rearrange the jobs according to non decreasing order of their due dates see [8], while the second step is to find the optimal solution by applying the following recursive equations:

$$\begin{aligned}
g(n_1,n_2,\ldots,n_f,\ldots,n_F,f) = \min\{ & \max\{\ g(n_1,n_2,\ldots,n_f-1,\ldots,n_F,f) + N\, p_{n_f f} - d_{n_f f}, 0\}, \\
& \min_{h\neq f}\{\max\{\ g(n_1,n_2,\ldots,n_f-1,\ldots,n_F,h) + N(s_f + p_{n_f f}) - d_{n_f f}, 0\}\}\} \qquad \ldots(13)
\end{aligned}$$

To explain the last recursive equations let us take the following example: consider a problem which consists of 4 jobs divided into two families f_1 and f_2 where $f_1=\{1, 3\}$ and $f_2=\{2, 4\}$ with set-up times $s_1=s_2=2$ of both families, the processing times the jobs are: $p_1=2$, $p_2=1$, $p_3=3$ and $p_4=8$ while the due dates of the jobs are $d_1=5$, $d_2=10$, $d_3=8$ and $d_4=19$. By applying the recursive equations (13) we will obtain the following table of solutions:

	(n_1, n_2)							
	(1, 0)	(0, 1)	(2, 0)	(0, 2)	(1, 1)	(1, 2)	(2, 1)	(2, 2)
$g(n_1,n_2,1)$	-1	∞	-1	∞	4	12	2	12
$g(n_1,n_2,2)$	∞	-5	∞	-5	1	1	2	2

Table 2

From the last column of the above table we see that the optimal solution is equal to 2, backtracking shows that the best sequence is (1, 3, 2, 4). We Notice that our dynamic algorithm to solve the maximum lateness problem with set-up times is equivalent to that of Ghosh and Gupta [4], since it required the same complexity time, that is of order $O(n^F)$.

4. Conclusions

In this paper we see that some special kinds of machine scheduling problems can be solved by applying dynamic programming algorithms with a fixed time, in other words all our dynamic algorithms are solvable in the same time of complexity. Some of these algorithms represent a generalization to other algorithms and some of them represent a modification of other ones while some of them are new ones. You can make use of this paper to solve other objective functions or to apply it to more than one machine.

REFERENCES

[1] Abdullah, K.A., (2005), Scheduling Two Job Classes on a Single Machine to Minimize The Total Weighted Completion Time, Al-Mustansiriya Journal Science 16, 3, 37-46.

[2] Abdul-Razaq, T.S, and Abdullah K.A., (2000), Scheduling Job Classes on Single Machine with Batches to Minimize The Sum of The Weighted Completion Times, Mustansiriya Journal Science, 11, 253-274.

[3] Ahn, B.H. and Hyun, J.H., (1990), Single Facility Multi-Class Job Scheduling, Computers and Operations Research, 17, 265-272.

[4] Crauwels, H.A.J., Hariri, A.M.A., Potts, C.N., and Van Wassenhove, L.N., (1998), Branch and Bound Algorithms for Single Machine Scheduling with Batch Set-up Times to Minimize Total Weighted Completion Time, Annals of Operations Research , 83, 59-76.

[5] Ghosh, J.B., (1994), Batch Scheduling to Minimize Total Completion Time, Operations Research Letters, 16, 271-275.

[6] Ghosh, J.B. and Gupta, J.N.D., (1997), Batch Scheduling to Minimize Maximum Lateness, Operations Research Letters, 21, 77-80.

[7] Graham, R.L. Lawler, E.L. Lenstra, J.K. and Rinnooy Kan, A.H.G., (1979), Optimization and approximation in deterministic machine scheduling: A survey, Annals of Discrete Mathematics 5, 287-326.

[8] Monma, C.L. and Potts, C.N., (1989), On The Complexity of Scheduling With Batch Set-up Times, Operations Research, 37, 798-804.

[9] Potts, C.N., (1991), Scheduling Two Job Classes on a Single Machine, Computers Operations Research 18, 5, 411-415.

[10] Potts, C.N. and Kovalyov, M.Y., (2000), Invited Review Scheduling with Batching: review, European Journal of Operational Research 120, 228-249.

[11] Townsend, W., (1978), The single machine problem with quadratic penalty function of completion times: A Branch-and-Bound solution, Management Science, 24, 5, 530-534.