Scheduling n jobs on a Single machine with multiple objectives and set-up times

Waffa Abdul-Abbas
Dept. of mathematics,
College of Education,
University of Thi-Qar.

المستخلص

تناولنا في هذا البحث دراسة مسألة جدولة n من النتاجات(jobs) مقسمه الى F من العوائل(families)على ماكنة واحدة. هدفنا في هذه الدراسة هو إيجاد الحل الأمثل والحلول التقريبية لجدولة النتاجات لتصغير دالة الهدف وهي المجموع ألوزني لأوقات الإتمام و المجموع ألوزني للنتاجات المتاخرة .

لحل هذه المسألة تم اشتقاق قيد أدنى (LB)بإرخاءشرط وجود وقت لإعداد (set-up time) وتجزئة المسألة الأصلية إلى مسألتين جزئيتين وذلك لاستخدامه في خوارزمية التفرع والتقيد والتي تستخدم لإيجاد الحل الأمثل.كما قمنا أيضا باستخدام مجموعة من الطرائق التقريبية Local search methods للحصول على حل قريب من الحل الامثل منها (Algorithm AH , Threshold Accepting, Tabu Search (TS)) .

Abstract

In this paper we consider the problem of scheduling n jobs on a single machine, where the jobs are divided into F families, each family f, (f = 1, …, F) contains n_f jobs. Our aim in this study is to find the optimal and near optimal schedule for the n jobs to minimize total weighted completion time and weighted number of tardy jops.

For solving this problem we propose a new lower bound (LB) based on relaxation of set-up times and decomposes the problem into tow sub-problems. To be used in a branch and bound

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

algorithm which used to find the optimal solution. We also applied some local search methods to find near optimal solution such as (algorithm AH, Threshold Accepting**,** Tabu search (TS)).

1-Introduction

In some setting, the grouping of jobs into families is a desirable or necessary tactic, because of the similarity of their production requirements [9].

No set up time is required for job if it belongs to the same family of the previously processed job. Otherwise it is necessary at the start of the schedule and on each occasion when the machine switches from a job in one family to a job in another family.

Various applications of family scheduling models are reported in literature. An example described by Conway, Maxwell and Miller (1967). [2], involves the production of different colors of paint on the same machine, a set up time is incurred for cleaning the machine whenever there is color change. This set up time depends on both: the color being removed and the color for which the machine being prepared.

In this case we talk about dependent set up time, that is dependent on both: the job to be processed and the immediately preceding one. A second example is the scheduling of computer system; it was described by Bruno and Downey [1]. We remark that the required set up time depends only on the time to load the compiler for the current job and does not depend on the previous job. Here we talk about the independent set up time.

2- Problem Formulation

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

To state our scheduling problem more precisely we are given: n jobs that are divided into F families, each family f, for f=1,2,3,…,F contains n_f jobs. The jobs are numbered 1, 2, …, n. Sometimes it is more convenient to refer to job (i, f), which is the i-th job in family f, for i=1,2,3,…,n_f .

All jobs are available for processing at time zero and to be scheduled on single machine. We let p_{if} denote the processing time of job (i, f), a positive weight w_{if} (importance weight for completion time of job (i, f), w'_{if} , a positive number (penalty for late of job (i, f)) and due date d_{if}.

Given a schedule, then for each job (i, f), we calculate the completion time of job (i, f) $c_i = \sum_{j=1}^{i} p_j$ and the zero one variable

where $\sigma(i)$ denote the position of job i in the ordering σ and S denoted the set of all sequences. The total cost for problem (p) consists of multiple objective of minimizing sum of weighted completion times $\sum_{i=1}^{n} w_i c_i$ and minimum number of weighted tardy jobs $\sum_{i=1}^{n} w'_i u_i$.

Using the three classifications suggested by Graham et. al [4], this problem is denoted by $1/s_f/\sum_{i=1}^{n} w_i c_i + \sum_{i=1}^{n} w'_i u_i$.

…… (P)

The problem (p) is NP-hard since the $1//\sum w'_i u_i$ problem is NP-hard. Since our multiple objective problem is NP-hard, we may find a sequence that gives minimum value for one of them but not both.

For this problem, there are the following cases:

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

* If there are no set up times (i. e. $s_f = 0$, f =1, …, F)and setting $w_i = w_i'$, (i=1, …, n) the resulting problem(P_1) is

$$Z = \sum_{i=1}^{n} w_{\sigma(i)} C_{\sigma(i)} + \sum_{i=1}^{n} w_{\sigma(i)} u_{\sigma(i)} \qquad \ldots\ldots\ldots (P_1)$$

s.t.

$$\begin{cases} C_{\sigma(i)} \geq P_{\sigma(i)} & i = 1,\ldots,n \\ C_{\sigma(i)} = C_{\sigma(i-1)} + P_{\sigma(i)} & i = 1,\ldots,n \end{cases}$$

Where

$$U_{\sigma(i)} = \begin{cases} 0 & \text{if} \quad C_{\sigma(i)} \leq P_{\sigma(i)} \\ 1 & \text{if} \quad C_{\sigma(i)} > P_{\sigma(i)} \end{cases}$$

The problem (P_1) is NP- hard since the $1//\sum_{i=1}^{n} w_i u_i$ problem is NP- hard. It is clear that the sequence used to solve the problem (P_1) obtained by (SWPT) rule usually performs not good when both objectives are to be considered.

* If there are no set up times (i. e. $s_f = 0$, f =1, …, F)and $w_i = w_i' = 1$, (i=1, …, n). hence the problem (P) is reduced to the problem (P_2)

$$Z = \sum_{i=1}^{n} C_{\sigma(i)} + \sum_{i=1}^{n} u_{\sigma(i)} \qquad \ldots\ldots (p_2)$$

s.t.

$$\begin{cases} C_{\sigma(i)} \geq P_{\sigma(i)} & i = 1,\ldots,n \\ C_{\sigma(i)} = C_{\sigma(i-1)} + P_{\sigma(i)} & i = 1,\ldots,n \end{cases}$$

Where

$$U_{\sigma(i)} = \begin{cases} 0 & \text{if} \quad C_{\sigma(i)} \leq P_{\sigma(i)} \\ 1 & \text{if} \quad C_{\sigma(i)} > P_{\sigma(i)} \end{cases}$$

Also still the problem (P_2) is not easy to be solved.

* If there are no set up times(i. e. $s_f = 0$, f =1, …, F)and setting ($w_i' = 0$, for(i=1, …, n))yields a polynomial algorithm to

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

minimize the total weighted completion time (i. e. $1//\sum w_i c_i$ problem) jobs should be sequenced in order of non-decreasing processing time (SWPT) schedule[8].

***** If (w_i =0 and w_i'=1, i=1, …, n)for(p), then to minimize the number of tardy jobs (i. e. $1//\sum_{i=1}^{n} u_i$ problem) jobs should be sequenced in order of non- decreasing due date, the earliest due date(EDD) schedule, and can be used Moore's algorithm(MA)[6].

Let $\sigma = (\sigma(1),..., \sigma(n))$ be the sequence obtained by ordering the jobs in (EDD) rule. In this sequence if there exists a job $\sigma(i)$ (with i as small as possible) that is completed after its due date, then one of the jobs sequenced in the first i positions and with largest processing time is selected to be late and removed from the set n.The procedure of (MA) continues until all the remaining jobs are completed by their due dates.

3- Lower and upper bounds

3-1 Lower bounds procedure:

In this section, we derive lower bound for our problem (p). At the root node of the search tree an initial lower bound (LB) on the cost of an initial schedule is obtained by relaxation of the set up times. We first relaxed the set up times for each family f(i. e. S_f=0, f=1, …,F), to get $1//\sum_{i=1}^{n} w_i c_i + \sum_{i=1}^{n} w_i' u_i$ problem. Now we decompose the problem into tow sub- problems with a simple structure. Then the lower bound of the problem (P) is the sum of the minimum value of the sub- problem (P_3) and the lower bound of the sub-problem (P_4) where,

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

$$Z_1 = \underset{\sigma \in s}{\text{Min}} \sum_{i \in n} W_{\sigma(i)} C_{\sigma(i)} \qquad \ldots\ldots (p_3)$$

s.t.

$$\begin{cases} C_{\sigma(i)} \geq P_{\sigma(i)} & i = 1,\ldots,n \\ C_{\sigma(i)} = C_{\sigma(i-1)} + P_{\sigma(i)} & i = 1,\ldots,n \end{cases}$$

Where

$$U_{\sigma(i)} = \begin{cases} 0 & \text{if} \quad C_{\sigma(i)} \leq P_{\sigma(i)} \\ 1 & \text{if} \quad C_{\sigma(i)} > P_{\sigma(i)} \end{cases}$$

$$\begin{cases} Z_2 = \text{Min} \sum_{i \in n} W'_{\sigma(i)} U_{\sigma(i)} \\ \text{s.t} \\ U_{\sigma(i)} = \begin{cases} 0 & \text{if} \quad C_{\sigma(i)} \leq d_{\sigma(i)} \\ 1 & \text{if} \quad C_{\sigma(i)} > d_{\sigma(i)} \end{cases} \end{cases} \qquad \ldots\ldots (P_4)$$

This decomposition has the following properties. First, (p_3) and (p_4) have simpler structure than (p), and thus appear easily to solve optimality for (p_1) (i. e. ((p_1) is solved in O(n log n) steps by (SWPT)rule). Second a lower bound can be obtained for (p_2), by using Moore's algorithm [MA].

Hence the lower bound LB is equal to the sum of $\sum_{i=1}^{n} w_i c_i$ for (p_1) and $\sum_{i=1}^{n} u_i$ number of late jobs for (p_2). This means that LB= $\sum_{i=1}^{n} w_i c_i + \sum_{i=1}^{n} u_i$.

3-2 The upper bound (UB) procedure:

The procedure begins by using two heuristic methods to schedule the jobs. The better of the two heuristic sequences is used to provide an initial upper bound (UB) at the root node of the search tree.

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

The first heuristic method ordered the jobs according to the shortest weighted processing time (SWPT) rule. (i. e. in non-decreasing order P_i/W_i (i=1, …, n)), then for this sequence we calculate the sum of weighted completion time and weighted number of tardy jobs and their value is (UB). The second heuristic method indexed the jobs within each family f(f=1, …, F) in (SWPT)order.

4- Branch and Bound (BAB) algorithm:

This section describes a (BAB) algorithm, which is the most widely solution technique that is used in scheduling problems. Initially, the procedure begins by using two heuristic methods of section (3-2) to generate an upper bound (UB) on the cost of an optimal schedule. Also, at the root node of the search tree an initial lower bound (ILB)on the cost of an optimal schedule is obtained from the procedure described in section(3-1). At any level, if a node has a lower bound greater than or equal to the current upper bound (UB) already computed, then this node is discarded. Otherwise, it may be selected for our next branching. The (BAB) method continues in a similar way by using a forward branching rule. Whenever a complete sequence is obtained, this sequence is evaluated and the upper bound (UB) is altered if the new value is less than the old one. The procedure is repeated until all nodes have been considered. Feasible solution with this (UB) is an optimal solution.

5- Local search for the problem (p):

It is clear that in solving scheduling problems one tends to use Branch and Bound (BAB) algorithm or Dynamic Programming

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

(DP) to find optimal solutions, however, these approaches have two main disadvantages:

- They are mathematically complex and thus a lot of times have to be invested.
- When it concerns NP-hard problem, the computation time requirements are enormous for large sized problem.

To avoid these drawbacks we can appeal to heuristic methods. In recent years, the improvement in heuristic methods becomes under the name 'local search heuristic' and implemented on the problem of scheduling(n) jobs on single machine to minimize total weighted completion time and weighted number of tardy jobs. the natural representation will be used to represent the solution. For each local search method a set of parameter setting is necessary for arriving at high performance algorithm.

First we introduce some neighborhoods for a permutation problem, where the step of feasible solutions is given by the set of permutation of (n) jobs [7].

- **Jump(JU)**: In a permutation $\sigma = (\sigma(1),..., \sigma(n))$ **,** select an arbitrary job $\sigma(i)$ and jump it to a smaller position j, i>j, or to a large position k, k>i . Thus, we have $|N(\sigma)| = (n-1)^2$.
- **Pairwise Interchange (PI)** In a permutation σ select two arbitrary jobs $\sigma(i)$ and $\sigma(j), i \neq j$ and interchange them, and $|N(\sigma)| = n(n-1)/2$.
- **Adjacent pairwise interchange (API):** This is a special case of the jump interchange neighborhood**.** In a permutation σ , two adjacent jobs $\sigma(i)$ and $\sigma(i+1), (i = 1,2,3,..., n-1)$ are interchanged to generate a neighbor σ , where $|N(\sigma)| = (n-1)$.

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

Now, we propose algorithm AH which is applied at local search methods to provide a best solution.

5-1 Algorithm AH [5]:

Step (1): (Initialization) to obtain an initial current solution the jobs are ordered according to the shortest weighted processing time(SWPT)rule, in which the jobs are sequenced in non-decreasing order of P_i/W_i (i=1, …, n) to obtain the current sequenced $\sigma_{\text{int}} = (\sigma(1),..., \sigma(n))$ with its objective function value of σ_{int} say f(σ_{int})= f_{int} .

Step (2): In this step the initial sequence σ_{int} will be changed by the others neighborhoods and function values are calculated for every one i. e.

a. For the neighbor Ju, have σ_{Ju} and f_{Ju} .

b. For the neighbor PI, have σ_{PI} and f_{PI} .

c. For the neighbor API, have σ_{API} and f_{API} .

Step(3): Now choose $\min f^* = \{f_{Ju}, f_{PI}, f_{API}, f_{Ini}\}$ and $\min \sigma^* = \{\sigma_{Ju}, \sigma_{PI}, \sigma_{API}, \sigma_{Ini}\}$, then set $f_{Ini} = min\ f^*$ and $\sigma_{Ini} = min\ \sigma^*$.

Step (4): (Termination) the algorithm is terminated after (500) iterations at a feasible solution.

5-2 Threshold Accepting (TA) method

Threshold Accepting (TA) method is similar to Simulated Annealing (SA) that uses deterministic acceptance rule for a solution that causes a deterioration in the objective value. Here a move is accepted providing that it doesn't increase the objective function by more than V where V is threshold value. Now, the TA method can be described as follows:

Step (1) :(Initialization) to obtain an initial current solution σ_{int} in the same way used in section (5-1).

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

Step(2) Using algorithm AH to generated new solution σ^*.
Step(3) In this step we are updating the threshold values V_k for $1 \le k \le l$ .We use $V_k = \xi V_{k-1}$ with $\xi = (V_l / V_1)^{\frac{1}{l-1}}$ where V_1 and V_l are initial and final threshold values respectively and set

$$\left.\begin{aligned} V_1 &= r_1 \ f(s_1) \\ V_l &= r_l \ f(s_l) \end{aligned}\right\}$$

where r_1, r_l are parameters that determine $r_1 = 0.02$, $r_l = 0.0001$ and s_1 is a feasible solution.

Step(4) (Termination) The algorithm is terminated after (500) iterations at a feasible solution.

5-3Tabu search (TS) method:

Glover [3] combines the deterministic iterative improvement algorithm with a possibility to accept cost increasing solution. In this way the search is directed a way from local minima, such that other parts of the solution space can be inspected. This is done by maintaining a finite list are not acceptable in next few iterations. This list is called the Tabu list. However, a solution on the Tabu list may be accepted if its quality is in some sense good enough, in which case it is said to attain a certain aspiration level.

Determination of (TS) algorithm parameters:

To determine the parameters of TS for the problem $1/s_f / \sum_{i=1}^{n} w_i C_i + \sum_{i=1}^{n} W_i' u_i$, we discuss each of the following issues:

Step (1) :(Initialization) To obtain an initial current solution σ_{int} in the same way used in section (5-1).

Step (2) :(Neighbourhood generated) the neighbor solution of the current solution is generated by the same procedure described in section (5-1) step (2).

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

Step (3): Add the new neighborhoods in the Tabu list if it is not existed, otherwise ignore it.

Step (4): (Termination) This termination condition used here is the same one described in section (5.1).

6-Computational experience

6-1 Test problems

The BAB algorithm was tested by coding it in Microsoft FORTRAN Power Station and the local search methods were tested by coding them in Matlab R2008a and run on a Pentium IV at 3.33 GHz, 512 MB computer.

A set of test problem was created to compare the performance of the algorithms. Forty random test problems were generated for different combination of numbers of jobs and families: the size of the test problems used are n=10, 20, 30, 50, 60, 70, 80 with F= 2, 5, 8. Jobs are distributed uniformly across families, so that each family contains either $\lfloor n/F \rfloor or \lceil n/F \rceil$ jobs. Processing times, weights and set-up times are randomly generated integers from the uniform distribution [1, 10]. An integer penalty w_i' was generated from the uniform distribution [1, 100].

We used the (BAB) algorithm described in section (4). The optimal solution of all 10-jobs and 20-jobs test problem with F=2, 5, 8 are obtained with reasonable limits on computation time. For problems that are not solved to optimality, the best solution was found by any of local search methods.

The local search methods that are described in section (5) were applied to all test problems and were compared.

6-2 Comparative results:

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

In this section we will report the results of the computational tests to show the effectiveness of the BAB algorithm and the local search methods. Twenty test problems for n=20 and F=4 are tested and the results are given in Table (1).

It is seen from Table (1) that each of the algorithms: algorithm AH, Threshold Accepting (TA), Tabu search (TS) generates solutions of high quality with respect to the difficulty of our objective function. It is clear from Table (1) that the best result as measured by the number of optimal solutions generated and the average deviations are obtained by Tabu search (TS), followed by Threshold Accepting and then algorithm AH.

Local search methods described in section (5) are compared in Table (2) which shows the importance of each method. An optimal solution to each test problem is obtained by (BAB) algorithm. Whenever a problem was not solved to optimality within a reasonable limit on computation time, the best solution value found by any of the local search methods forms the basis for comparison.

The local search methods are compared by listing for each value of n the number of times out of 40 that an optimal solution is found (NO).

Number	**BAB**	**AH**	**TA**	**TS**
1	**99251**	**99251***	**99941**	**99251***
2	**73539**	**80609**	**78734**	**74785**

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

3	79388	84382	79388*	79946
4	96954	99767	99359	98906
5	73199	73837	73199*	73199*
6	91272	91272*	93358	91272*
7	82387	87948	86413	87227
8	78177	79436	78177*	78177*
9	93291	99106	95820	94128
10	50474	59002	57034	50474*
11	77556	77557*	77557*	77557*
12	83219	87055	83966	83219*
13	97580	98515	97580*	97580*
14	78257	78257*	78838	78257*
15	78458	79717	79035	79598
16	75129	76659	75129*	77663
17	77893	77893*	78761	77893*
18	65691	68013	67393	67296
19	68769	69372	68791*	68791*
20	63932	63923*	63923*	63923*

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

The results in Table (2) shows that (TS) method performs very well, especially for the large problem instances, but it is computationally time consuming. Also it is clear from Table (2) that (AH) and (TA) gave reasonable results.

The main conclusion to be drawn from the computation results is that (TS) is more effective method for our problem. For the large problem instances (TA) and (AH) have generated a good quality solution.

Finally, all heuristics presented in this paper generate quite good solutions to our problem with respect to the difficulty of our objective function.

Table (1). Comparison between local search method values and optimal solution.

***** Indicates that the value of heuristic method is equal to the optimal value.

BAB: Branch and Bound method.

AH: The algorithm AH.

TA: Threshold Accepting method.

TS: Tabu Search method.

Table (2). Comparative results

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

N	F	AH			TA			TS		
		NO	AD	ACT	NO	AD	ACT	NO	AD	ACT
	2	17	70	0.32	19	82	0.41	21	47	0.63
10	5	10	186	0.38	13	131	0.46	16	74	0.62
	8	23	164	0.44	25	45	0.44	28	56	0.74
	2	21	320	0.50	24	221	0.59	18	216	0.98
20	5	12	304	0.27	12	128	0.72	8	338	1.33
	8	15	209	0.27	17	332	0.74	11	145	1.98
	2	13	613	0.43	10	302	0.50	20	315	1.40
30	5	15	998	0.66	12	465	0.68	23	328	1.83
	8	14	1274	0.71	11	864	0.79	25	454	1.46
	2	9	5344	0.33	10	1139	0.83	17	357	2.32
50	5	11	5846	0.23	6	1674	0.91	31	276	2.32
	8	13	4556	0.17	5	1431	0.55	24	987	3.27
	2	8	7654	0.66	3	3530	1.32	34	368	1.78
60	5	6	7562	0.28	4	2318	1.45	27	573	2.46
	8	2	8725	0.09	4	4238	1.66	39	235	2.80
70	2	4	9893	0.22	6	5456	1.73	36	783	3.11

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

	5	**3**	**14168**	**0.23**	**4**	**4367**	**1.76**	**40**	**0**	**4.45**
	8	**4**	**15425**	**0.22**	**7**	**3473**	**2.00**	**40**	**0**	**3.23**
	2	**0**	**17452**	**0.28**	**3**	**8227**	**2.20**	**35**	**524**	**4.84**
80	**5**	**0**	**18357**	**0.35**	**2**	**3626**	**2.20**	**39**	**243**	**4.79**
	8	**0**	**15460**	**0.67**	**3**	**6382**	**2.72**	**40**	**0**	**4.62**

N: Number of jobs.

F: Number of families.

NO: Number of times out of 40 that an optimal solution is found.

AD: The average deviation.

ACT: The average computation time in seconds.

TA: Threshold Accepting method.

TS: Tabu Search method.

References:

1. Bruno J. and Downey P., Complexity of task sequencing with deadlines, set-up times and changeover costs, SIAM Journal on computing 7, (1978) 393-404.
2. Conway R. W., Maxwell W. L. and Miller L. W., Theory of scheduling, Addison Wesley, Reading, MA, (1976).
3. F. Glover. Tabu search part I. ORSA Journal on Computing. 1:190-206, 1989.
4. Graham R. L, Lawler E. L, Lenstra J. K, Rinnooy Kan A. H. G., Optimization and approximation in deterministic sequencing and scheduling. A survey Ann Discrete Math, 5, 287-326(1979).

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional

5. H. A. Mohammed, H. A. Cheachan and Q. A. Khtan, Single machine scheduling to minimizing sum penality number of late jobs subject to minimize the sum weight of completion time. Journal of Kerbala University. 7(1), 2009.
6. Moore J.M., "An n job, one machine sequencing algorithm for minimize the number of late jobs", Management Scince 15, 102-109, (1968).
7. R. Ruis and T stutzle, Simple and effective iterated greedy algorithm for permutation flowshop scheduling problem. European Journal of operation research. 177(3):2033-2049, 2007.
8. Smith W. E., "Various optimizers for single-stage production", Naval Res. Logist. Quart. 3, 59-66, (1956).
9. Webster S., Baker K. R. Scheduling group of jobs on a single machine. Oper. Res. Vol. 43, 692-703(1995).

Created with nitroPDF professional
download the free trial online at nitropdf.com/professional