Real-World Extensions to a Production Scheduling Algorithm Based on Lagrangian Relaxation

Y. Narahari
Computer Science and Automation
Indian Institute of Science
Bangalore 560 012 - INDIA

Sundar Ram Vedula
Mechanical Engineering
Indian Institute of Technology
Madras 600 036 - INDIA

Abstract

In this article, we develop three efficient extensions to a recent production scheduling algorithm based on Lagrangian relaxation [9]. These extensions handle the following real-world features:

- *stochastic availability of raw material,*
- *reworking and scrapping of parts, and*
- *incorporating set-up costs and set-up times in the scheduling of multiclass manufacturing systems*

The first two extensions yield on-line algorithms which are illustrated using a 89-job scheduling problem. The third extension uses an algorithmic procedure to insert switchovers to obtain a good schedule and is illustrated for a two-class, 89-job scheduling problem.

1 Introduction

Discrete activity scheduling [2, 3, 7, 12] is probably the most important issue in the design and operation of manufacturing systems. This paper is concerned with deterministic machine scheduling in a system comprising parallel identical machines. We present three real-world extensions to a recent production scheduling algorithm based on Lagrangian relaxation [9]. The extensions handle the following real-world features:

1. Stochastic availability of raw material
2. Reworking and scrapping of parts
3. Switchover costs and set-up times encountered in the scheduling of multiclass manufacturing systems.

These extensions are illustrated for a detailed 89-job example adapted from a real-world problem presented in [9].

1.1 Schedule Generation Using Lagrangian Relaxation

The theory of sequencing and scheduling, more than any other area in Operations Research, is characterized by a virtually unlimited number of problem types [2, 7]. Recently, in the area of deterministic machine scheduling, scheduling algorithms based on Lagrangian relaxation [1, 8] have been proposed and have proved to be highly effective in real-world factory situations. These Lagrangian relaxation-based algorithms have been reviewed in [6]. Luh, Hoitomt, Max, and Pattipati [9] presented the first of these algorithms for the case of scheduling single operation jobs on parallel identical machines. Later, the approach was extended to the case of multiple-operation jobs with precedence constraints [5]. More recently, the approach has been extended to general job shop scheduling problems [4, 6]. The algorithms use a decomposable objective function such as the total weighted tardiness and obtain the solution by solving independent, job-level subproblems.

We now give a brief description of the basic technique that we employ in this paper. This technique is due to Luh, Hoitomt, Max, and Pattipati [9]. The method is used for the nonpreemptive discrete time scheduling of independent jobs on identical, parallel machines and is based on the Lagrangian relaxation algorithm [1,8]. The input variables are:
N : Number of jobs
K : Time horizon for scheduling
$w_1, \cdots w_N$: Weights (relative importance) of jobs
$t_1, \cdots, t_N$: Processing times of the jobs
$D_1, \cdots, D_N$: Due dates of the jobs
$M_1, \cdots, M_K$: Numbers of machines available at discrete time instants $1, 2, \cdots, K$, respectively.

The decision variables are $B_1, \cdots, B_N$, the beginning times for the jobs. Now, define, for $i = 1, \cdots, N$, and $k = 1, \cdots, K$,
$\delta_{ik} = 1$ if job i is undergoing processing at time k

$\delta_{ik} = 0$ if job i is not undergoing processing at time k
C_i = Completion time of job i
$T_i = max\{0, C_i - D_i\}$

T_i represents the tardiness of job i. The objective function of interest is

$$J = \sum_{i=1}^{N} w_i T_i$$

which is the weighted tardiness criterion. This criterion takes into account the relative importance of jobs, and the importance of meeting due dates. The scheduling problem can now be stated as follows.

$$\min_{\{B_i\}} J \quad \text{with} \quad J = \sum_{i=1}^{N} w_i T_i$$

subject to capacity constraints:

$$\sum_{i=1}^{N} \delta_{ik} \leq M_k, k = 1, \cdots, K$$

and processing time requirements:

$$C_i - B_i + 1 = t_i, i = 1, \cdots, N$$

The problem is solved by maximizing the dual function which is obtained by using a set of multipliers to relax the capacity constraints. The dual problem can be decomposed into N subproblems, one for each job. The computational complexity of each subproblem is linear in K. Using these N solutions, the dual problem is solved using a subgradient method [10]. The stopping criterion used in the above subgradient method does not always yield a feasible schedule, in the sense that the capacity constraints might be violated for a few time slots [9]. To construct a feasible schedule, a greedy heuristic based on the list scheduling concept is used.

1.2 Extensions Proposed

In this paper, we consider the LR (Lagrangian relaxation) approach presented in [9] and develop extensions to address three real-world features in scheduling. These features are diffferent from the extensions to the algorithm presented in [9].

The first extension proposed handles delayed or stochastic availability of raw material. Since raw materials are usually procured from sources external to the machine shop, one is never sure unless the raw material is in hand. The LR algorithm of Luh *et al* [9] handles in a limited way the feature of delayed availability of raw material, by specifying earliest start times, which are however assumed to be known in advance. In a stochastic environment where the raw material keeps arriving into the shop at time instants not known in advance, the earliest start times cannot capture fully the situation.

The second extension seeks to address the rescheduling problem that arises when processed parts are either rejected or sent for reworking after periodic inspections. Parts identified for reworking cause extra load on the system whereas every rejected part entails complete reprocessing and also material waste. As a result, the originally drawn schedule may not be the best any more and rescheduling becomes necessary.

The problem of set-ups in multiclass manufacturing system is addressed by the third extension we propose. Our method enables set-up costs to be included in the scheduling procedure. For this, we first obtain a schedule without set-up operations (using the LR-algorithm [9]) and modify this schedule using a simple heuristic algorithm, to include the effect of set-up times and set-up costs.

The paper is organized as follows. In Section 2, we present the input data for a 89-job scheduling problem, which is used for illustrating the three extensions proposed. This data is adapted from a real-world situation described in [9]. The three extensions and relevant numerical experimentation are presented in Sections 3, 4, and 5.

2 Data for Scheduling Example

Table 1 shows the input data for a 89-job system. This data is the same as in [9] and is a real-world data taken from the tool and die workcenter of Pratt and Whitney's Development operations shop. We assume that the machine shop has 4 identical machines (the data in [9] has 42 machines). The first column shows the job number; the second column, the job weight; the third column, the job processing time; and the fourth column, the job due date. The fifth column is a job class column introduced in this paper, specially for discussing the set-up problem in Section 5. In the three sections that follow, we apply the proposed extensions to the above data by introducing specific changes needed.

i	w_i	t_i	D_i	cl	i	w_i	t_i	D_i	cl
1	1	1	0	A	2	1	2	15	A
3	9	1	12	A	4	1	1	13	A
5	1	1	1	A	6	1	1	-3	A
7	1	2	-2	A	8	1	4	0	B
9	1	3	9	B	10	1	1	7	A
11	1	1	8	A	12	1	2	13	A
13	6	5	19	B	14	1	1	2	A
15	1	1	7	A	16	1	2	-1	A
17	5	1	1	A	18	1	3	0	B
19	1	1	5	A	20	9	10	9	B
21	1	1	1	A	22	1	2	8	A
23	1	3	0	B	24	1	7	3	B
25	6	8	60	B	26	1	9	8	B
27	1	2	-1	A	28	1	3	3	B
29	1	2	0	A	30	6	1	2	A
31	9	1	-1	A	32	1	1	7	A
33	1	11	11	B	34	1	2	15	A
35	1	16	24	B	36	16	5	2	B
37	1	3	25	B	38	1	3	24	B
39	1	3	20	B	40	1	4	18	B
41	1	1	2	A	42	1	3	2	B
43	6	5	10	B	44	1	3	7	B
45	1	1	7	A	46	1	3	2	B
47	1	2	7	A	48	1	1	7	A
49	1	1	8	A	50	1	2	7	A
51	1	2	7	A	52	1	1	10	A
53	1	1	9	A	54	1	3	2	B
55	9	4	1	B	56	1	3	9	B
57	1	3	2	B	58	1	2	-1	A
59	1	1	10	A	60	1	20	24	B
61	1	5	5	B	62	1	4	9	B
63	1	6	7	B	64	16	5	5	B
65	1	1	10	A	66	1	3	2	B
67	1	3	7	B	68	1	15	42	B
69	1	8	10	B	70	1	16	17	B
71	1	10	20	B	72	1	3	-1	B
73	1	8	11	B	74	1	4	4	B
75	1	4	15	B	76	1	3	5	B
77	1	3	2	B	78	1	3	1	B
79	1	1	-1	A	80	6	3	12	B
81	1	3	10	B	82	1	3	9	B
83	1	2	4	A	84	1	3	2	B
85	6	12	9	B	86	1	2	1	A
87	1	2	1	A	88	1	2	1	A
89	1	2	0	A					

Table 1: Detailed data for scheduling example

3 Stochastic Availability of Raw Material

The LR scheduling algorithm of [9] assumes that raw material for the jobs to be scheduled is available when required. Since raw materials are usually procured from sources external to the machine shop, one can never be sure unless the raw material is in hand. The algorithm in [9] does handle the feature of delayed availability of raw material by specifying the earliest start times for individual jobs. But the earliest start times are specified in the very beginning when the algorithm is run. Consequently, the earliest start times cannot completely capture a stochastic environment where the raw material keeps arriving into the shop at time instants not known in advance. We propose the following method.

Let S be the set of all jobs. Assume that $T_0, T_1, \cdots, T_n$ are the time instants during the time horizon K at which raw material for jobs arrives into the machine shop. The instants $T_0, T_1, \cdots, T_n$, are not known in advance. Let $S_0, S_1, \cdots, S_N$ be the sets of jobs for which the raw material becomes available at $T_0, T_1, \cdots, T_n$, respectively. Assume further that $T_0 = 0$ so that S_0 is the set of jobs for which raw material is available at time zero. Table 2 shows a typical scenario for the 89-job example. The set S_0 in the table contains all the jobs not belonging to the sets S_1, S_2, S_3, and S_4.

Time instant	Value of T_i	S_i
T_0	0	S_0
T_1	10	{5,9,21,30,34,41,63 }
T_2	25	{62,76,84 }
T_3	28	{3,43,65,81,82 }
T_4	100	{13,25,68,71,75,80 }

Table 2. Data for delayed availability of raw material

Since the instants $T_1 = 10, T_2 = 25, T_3 = 28$ and $T_4 = 100$, are real-time instants which are not known in advance, they cannot be adequately specified by earliest start times. At time $T_0 = 0$, we first schedule, using the LR algorithm, all jobs in S_0, to obtain beginning times $\{B_i\}$ for all $i \in S_0$. The production commences and jobs in S_0 are scheduled according to $\{B_i\}$. At the instant T_1, raw material for jobs in S_1 becomes available, so there is a need to obtain an updated schedule that includes all jobs in S_1. At time T_1, the set S_0 can be partitioned into the following three sets.

$$S_{01} = \{i \in S_0 : B_i < T_1 \ \ and \ \ B_i + t_i \leq T_1\}$$

$$S_{02} = \{i \in S_0 : B_i < T_1 \ \ and \ \ B_i + t_i > T_1\}$$
$$S_{03} = \{i \in S_0 : B_i \geq T_1\}$$

Note in the above that B_i and t_i are the beginning time and processing time, respectively of job $i (i \in S_0)$. Jobs belonging to S_{01} would already be processed by T_1 and pose no problem. Jobs belonging to S_{02} would be undergoing processing at time T_1 whereas jobs in S_{03} are not even initiated by time T_1. The rescheduling that we need to do will therefore include all jobs in S_{03} and S_1 and possibly S_{02}. Whether or not we reschedule jobs in S_{02} depends on whether we prefer to preempt the jobs in S_{02} at time T_1 or we prefer to wait until the instant at which all jobs in S_{02} are processed. In the former case, the jobs rescheduled belong to the set $S_{02} \bigcup S_{03} \bigcup S_1$, the rescheduling is done starting from $t = T_1$, and certain machine time and possibly materials are wasted. In the latter case, the jobs rescheduled belong to the set $S_{03} \bigcup S_1$, the rescheduling is done at $t = T_1 + T_{02}$ where T_{02} is the time needed for completing the jobs already initiated, and a higher weighted tardiness is to be expected. The rescheduling is again done using the LR algorithm of [9]. The above procedure is repeated at the time instants $T_2, T_3, \cdots, T_n$.

3.1 Numerical Example

Consider the raw material arrival data given in Table 2. Recall that the 89 jobs specified in Table 1 have to be scheduled on to four machines. We first obtain the schedule, using the LR algorithm [9], for all jobs in the set S_0 (see Table 2). At time $t = 10$, raw material for jobs in S_1 arrives and rescheduling becomes necessary. Assuming that we wait until all jobs that are already initiated at time $t = 10$ to be completed, the rescheduling is done only at $t = 18$.

M_1	79	58	86	50	22	12	18	34	46
	54	9	40	64	79	73	3	65	80
	75								
M_2	55	1	7	21	41	15	59	29	4
	87	83	47	51	23	42	66	76	37
	74	33	43	13	68				
M_3	31	17	6	14	85	16	27	63	5
	19	10	48	11	52	88	2	77	44
	67	56	38	24	70	60	35	82	25
M_4	36	20	30	32	45	49	53	89	72
	78	57	84	28	39	8	62	61	26
	81	71							

Table 3: Final sequencing of jobs taking into account delayed availability of raw material

Table 3 shows the final schedule for all the jobs taking into account the raw material arrival sequence in Table 2. From Table 3, we note the following. Jobs 30 and 63 which belong to S_1 are scheduled immediately due to their high weights. Jobs 62, 76, and 84 in S_2 are found to be delayed because of their late due dates. Since raw material for jobs in S_4 arrive only at $t = 100$, the four machines will have to be idle starting from $t = 85$ onwards, until $t = 100$.

4 Scrapping and Reworking of Jobs

Quality control considerations entail inspection of processed jobs at frequent intervals to verify if the quality is acceptable. A typical inspection operation can have three outcomes: accept, reject, or rework. In case it is decided that a finished part is to be rejected or reworked, rescheduling becomes necessary. We propose the following scheme to handle such situations.

Assume that inspection operations are carried out at time instants $T_1, T_2, \cdots, T_n$. At time T_1, all parts processed until that time are inspected and parts to be rejected and parts to be reworked along with reworking times are decided. Rescheduling is done using the LR algorithm so as to include the parts identified for rejection and reworking. The due dates and weights of these parts can be assumed to be the original ones. If S is the set of all jobs; S_1, the set of jobs processed by time T_1; and S_r the set of jobs identified for rejection or reworking at T_1, then the set of jobs rescheduled at T_1 will be $(S - S_1) \bigcup S_r$. The procedure is repeated at all inspection epochs and we thus have an on-line procedure to handle scrapping and reworking of jobs.

4.1 Numerical Example

Table 4 shows a particular inspection program for the 89-job system of Table 1. There are three inspection epochs, T_1, T_2, and T_3. At the epoch $T_1 = 20$, all jobs processed until then are inspected and it is found that job 16 is to be rejected and job 20 is to be reworked. Thus the reworking time for job 16 is 2 (original processing time = 2). The reworking time for job 20 is decided to be 7 (original processing time = 10). The rescheduling done at this epoch will therefore include these two jobs. Similarly, rescheduling is done as per the data in Table 4. Table 5 shows the overall schedule for all the 89 jobs, taking into account the scrapping and reworking. We find that job 20 is reworked immediately whereas job 16 is postponed by a significant amount.

Epoch	i	Time
20	16	2
20	20	7
50	9	3
50	49	4
50	17	1
50	23	3
150	10	6
150	35	5
150	60	8
150	26	9

Table 4: Data for scrapping and reworking of jobs

M_1	36	79	14	43	52	4	7	13	27
	29	87	22	42	67	7	**17**	38	**9**
	49	62	**16**	73	26	**35**			
M_2	55	6	1	10	32	48	16	58	88
	83	51	12	18	84	76	44	81	37
	74	75	61	69	33	68	**60**		
M_3	31	30	41	19	15	45	11	49	53
	59	3	**20**	89	2	85	23	46	77
	56	82	**23**	8	40	64	70	35	**10**
M_4	17	63	5	21	20	65	80	86	47
	50	34	72	78	54	57	66	28	39
	25	24	60	**26**					

Table 5: Final sequencing taking into account reworking and scrapping of jobs. Bold faced jobs indicate scrapped or reworked parts.

5 Effect of Set-ups in a Multiclass Production System

In a multiclass production system, switchover times or set-up times can have a significant effect on the way parts are scheduled. Set-up costs would add a new dimension to the scheduling problem addressed by the LR algorithm [9] and incorporating set-up costs in the objective function will make the approach non-decomposable. Furthermore, the subproblems are not mutually independent any more, as jobs processed between successive set ups necessarily belong to the same class.

Assuming identical parallel machines to be available, we propose the following methodology for incorporating set up operations in the final schedules. First, the LR algorithm is used to schedule the given set of jobs in the available machines. This schedule will give a particular sequence of jobs on each machine. Let $J_{x_1}, J_{x_2}, \cdots, J_{x_p}$ be the sequence of jobs obtained for a specific machine. Let us assume these jobs belong to C different classes and when the machine switches over from one job class to another, it incurs a set-up cost and also a set up time. The schedule $J_{x_1}, \cdots, J_{x_p}$, is nearly optimal if set-ups are not included. However if the same sequence is used and we include the set up costs, the resulting schedule can be very inefficient due to frequent changeovers. Our idea is to group the jobs in the sequence $J_{x_1}, \cdots, J_{x_p}$, in a suitable way, without disturbing the sequence, but at the same time minimize the total cost which will be the sum of the weighted tardiness and set-up cost.

We illustrate our approach by viewing the 89-jobs of Table 1 as belonging to two different classes, say A and B. The class information is shown in Table 1. We have used the following criterion for classifying the jobs: jobs with processing time 1 or 2 are grouped as class A jobs and the rest of the jobs are designated as class B jobs. We also assume that the switchover time from A to B is 6 units and that from B to A is 4 units.

Initially, the 89 jobs are scheduled using the LR algorithm onto the 4 machines. Let us look at the schedule for a particular machine. Let $J_{a_1}, J_{a_2}, \cdots, J_{a_{n_a}}$ be the class A jobs in this schedule (in the same sequence as they appear in the schedule). Similarly, let $J_{b_1}, J_{b_2}, \cdots, J_{b_{n_b}}$ be the sequence of class B jobs in the schedule. First, we determine a range of suitable values for the number of switchovers using considerations such as:

1. Too many switchovers will make the set up costs dominate over the tardiness costs.
2. Too few switchovers will make the tardiness costs of delayed jobs substantial.
3. In a given set up, it is a good idea to process at least as many jobs having a total processing time equal to the set-up time.

Having chosen a certain number of switchovers, say s, we first look at the schedule having equal batch sizes for class A jobs and equal batch sizes for class B jobs. For example, in the above case, the batch size for class A is chosen as $\frac{2n_a}{s}$ and that for class B as $\frac{2n_b}{s}$. Calling q_a and q_b as the batch sizes for A and B, the sequence will look like one of the following:

$$J_{a_1}, \cdots, J_{a_{q_a}}, \text{switch}, J_{b_1}, \cdots, J_{b_{q_b}}, \text{switch}, \cdots$$

$$J_{b_1}, \cdots, J_{b_{q_b}}, \text{switch}, J_{a_1}, \cdots, J_{a_{q_a}}, \text{switch}, \cdots$$

The objective function for such a sequence can be easily evaluated. The position of *switch* can now be changed by one or more positions to the left or right in a systematic way and in each case, the total cost of the resulting schedule can be evaluated. The procedure can be repeated until changes in the total cost

keep decreasing and become negligible. This methodology can be repeated for other appropriate values of s (number of switchovers). Schedules with set-ups can be obtained for other machines in an identical way.

5.1 Numerical Example

For the input data in Table 1, an LR schedule was obtained for each of the four machines. Assuming a switchover time of 6(4) units for a change from class A (class B) to class B (class A), a schedule with set-ups was obtained for each of the four machines. Table 6 shows the resulting schedules.

M_1	36	43	6	13	21	16	58	86	83
	51	3	12	72	78	57	28	56	39
	74	24	71	60					
M_2	30	79	5	14	41	10	32	11	48
	55	85	13	54	84	67	81	8	40
	69	26	35						
M_3	31	17	19	15	45	49	63	80	18
	42	66	76	9	38	29	87	47	22
	59	2	62	61	33	70			
M_4	48	53	7	27	89	88	50	52	65
	34	20	23	46	77	44	82	37	75
	64	73	25	68					

Table 6: Final sequencing taking into account set-ups

6 Summary and Future Work

Three extensions have been proposed in this paper to the Lagrangian relaxation based scheduling algorithm of Luh, Hoitomt, Max, and Pattipati [9], to address real-world features such as stochastic availability of raw material, scrapping and reworking of parts, and existence of significant set-up costs. The extensions proposed can be implemented easily. Numerical experiments on a 89-job system have shown that these real-world features can often alter the original schedules in a significant way.

The issue of addressing the set-up problem needs to be examined in more detail. What we have provided is an efficient scheme for scheduling a two class production system with set-up times. The method needs to be extended to a multiclass system. A hybrid methodology that uses simulated annealing and Lagrangian relaxation has recently been developed to handle the set-up problem in systems with three or more job classes [11]. Other important questions in this context that need to be explored are: How to choose classes? How to choose batch sizes? Also, future work is needed to explore these extensions to multi-operation models and job shops.

References

1. M.L. Fisher, Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, Volume 27, 1981, pp. 1-18.
2. S. French, *Sequenching and Scheduling*. Wiley, New York, 1982.
3. S.B. Gershwin, Hierarchical Flow Control: A Framework for Scheduling and Planning Discrete Events in Manufacturing Systems. *Proceedings of the IEEE*, Volume 77, Number 1, January 1989, pp. 195-209.
4. D.J. Hoitomt, P.B. Luh, and K.R. Pattipati, Job Shop Scheduling. *Proceedings of First International Conference on Automation Technology*, Taipei, Taiwan, July 1990, pp. 565-574.
5. D.J. Hoitomt, P.B. Luh, E. Max, and K.R. Pattipati, Scheduling Jobs with Simple Precedence Constraints on Parallel Machines. Chapter 20, in: Y.C. Ho (Editor), *Discrete Event Systems*, IEEE Press, 1991.
6. D.J. Hoitomt, P.B. Luh, and K.R. Pattipati, A Practical Approach to Job Shop Scheduling Problems, To appear in: *IEEE Transactions on Robotics and Automation*, 1993.
7. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, Sequencing and Scheduling: Algorithms and Complexity. *Report No. BS- R8909*, Centre for Mathematics and Computer Science, Amsterdam, November 1989.
8. D.G. Luenberger, *Linear and Non-linear Programming*. Second Edition: Addison-Wesley, Reading, Massachusetts, 1984.
9. P. Luh, D. Hoitomt, E. Max, and K. Pattipati, Schedule Generation and Reconfiguration for Parallel Machines. *IEEE Transactions on Robotics and Automation*, Volume 6, Number 6, December 1990, pp. 687-696.
10. B.T. Polyak, Minimization of Unsmooth Functionals. *USSR Computational Mathematics and Mathematical Physics*, Volume 9, 1969, pp. 14-29.
11. R. Srigopal, *Scheduling Multiclass Production Facilities using Lagrangian Relaxation*. Master's Thesis, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, January 1994.
12. N. Viswanadham and Y. Narahari, *Performance Modeling of Automated Manufacturing Systems*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.