

Graph Decomposition Heuristic for Machine Scheduling Problems

Eui-seok Byeon

Department of Industrial Engineering, Sun Moon University
#100 Kalsanri, Tangjeongmyeon, Asansi, Chungnam 336-708, Korea

<Abstract>

A graph decomposition technique has been used for the application of machine scheduling problems. The method divides a problem with a series of sub-problem by solving partitioning problems. These sub-problems are turned out to be mutually exclusive and exhaustive assignment problems. The decomposition methodology makes only a part of the scheduling decisions at first while leaving the rest of the decisions to be made dynamically. This method retains flexibility in the schedule which can be used to improve its robustness. The proposed method can indeed handle scheduling disturbances more effectively when compared with traditional methods.

1. Introduction

Machine scheduling problems addresses one of the most critical issues in modern manufacturing systems. The Job-shop scheduling problem (JSP) for weighted tardiness is well known to be NP-hard in a strong sense (Ullman, 1975). Exact solutions for practical-size problems will be highly inefficient. Scheduling heuristics, therefore, are typically found in the literature [c.f. Morton et al. (1988), Roundy et, al. (1991)]. If we consider real-life situations such as machine breakdowns, rush orders, power failure, and other distributions, the existing scheduling methods will be even more limited.

In this paper, we present a new method for weighted tardiness schedule using a graph theoretic decomposition heuristic. This heuristic decomposes a given JSP into a series of sub-problem by solving a variant of the assignment problem (VAP). The assignment defines, in turn, a partially solved job-shop schedule that retains a great deal of local flexibility, and a much-reduced complexity. The local flexibility offers additional opportunities for the schedule to handle possible disturbances in the systems. Thus, a non-myopic view of the scheduling function is maintained throughout planning, while the attained local flexibility allows timely responses to shop disturbances.

Roundy et al. (1991) have also addressed the issue of scheduling flexibility. They reformulated Pritsker's (1968) Integer Programming formulation and implemented it as a two-phase approach including in a planning and a dispatching module. The planning module calculates the price of using each machine over the planning horizon with the dispatching module schedule based the price. Their approach is designed to handle minor system disturbances. Bean et al. (1991) develop a matchup scheduling algorithm, which computes a transient schedule upon the occurrence of a system disruption. The schedule eventually matches up with the pre-schedule in a finite amount of time while assuming an infinite planning horizon.

Morton et al. (1988), Lawrence and Morton (1993) and Vepsalainen and Morton (1987) have developed dispatching heuristics based on look-ahead pricing schemes. These heuristics were shown to be quite effective for realistic size problems. In the study, we have shown that when implemented in an iterative fashion, the performance of these heuristics can be much improved.

Sidney (1975) proposed an optimal decomposition procedure for minimizing weighed flow time in single machine problems with precedence constraints. The problem is decomposed into subsets based on the precedence network. Nonetheless, the number of subsets to be considered is not polynomially bounded by the problem size. Potts and Van Wassenhove (1982) have applied Lawler's decomposition theorem (1977) to solve a one-machine total tardiness problem. The author stated, however, that the approach was limited to that objective and could not be generalized to

other scheduling problems. Pinedo and Singer(1999) decomposed JSP into a number of single-machine subproblems that are solved one after another. This method is shown to be very sensitive to the order of subproblems with respect to quality and time.

2. Problem Statement

Our variant of the assignment problem (VAP) is defined by reference to the disjunctive graph associated with a job shop scheduling problems. We adopt the following notation:

N	a set of operation to be scheduled, including a dummy source, o, and sink, *.
A	a set of conjunctive arcs representing precedence constraints.
E	a set of disjunctive arcs representing operations which may compete for the same machine.
E_m	a set of disjunctive arcs for machine m.
α_k	number of operations to be assigned to subset k.
O	the set of last operations of each job.
X_{ik}	a binary variable equals to 1 operation i is assigned to subset k, 0 otherwise.
w_j	weight of job j.
dd_j	due date of job j.
r_i	ready time of operation i.
P_i	processing time of operation i.
t_i	actual starting time of operation i.

Given a job shop scheduling problem and its associated disjunctive graph $G(N,A,E)$, (Adams et al. 1988) we first assign each of the $|N|$ operations into p subsets. This assignment problem, or VAP can be expressed mathematically as follows:

$$(VAP): \text{Minimize } \sum_i \sum_k c_{ik} X_{ik} \quad (1)$$

s.t

$$\sum_k X_{ik} = 1 \quad i \in N \quad (2)$$

$$\sum_i X_{ik} = 1 \quad k=1,\dots,p \quad (3)$$

$$X_{jl} \leq \sum_{k=1}^l X_{ik} \quad (i,j) \in A \text{ and } l=1,\dots,p \quad (4)$$

$$X_{ik} \in \{0,1\} \quad i \in N \text{ and } k=1,\dots,p \quad (5)$$

Constraints (2) and (5) would be familiar with a set partitioning problem. The precedence constraints with respect to the job-shop scheduling problem (JSP) are represented in (4) by stating that operation j subject to $(i,j) \in A$ can be assigned only if its predecessor i has already been assigned to the current or an earlier subset. We term (4) the partial ordering constraints. The problem with (2), (3) and (5) is known as a generalized assignment problem (Ross and Soland, 1975). Note that solution of VAP will resolve some disjunctive arcs in the set E. In general, an arc (i,j) with i,j not assigned to a same subset will have its direction fixed. Suppose \overline{E}_m^k is the remaining, unresolved set of

disjunctive arcs in subset k , on machine m , then $\overline{E}_m^k = \{(i,j) \in E_m \mid x_{ik} = x_{jk} = 1\}$. The resolved set of disjunctive arcs on machine m may be expressed as,

$$\sigma_k = \{(i,j) \in E_m \mid x_{ik} = x_{jk} = 1 \text{ and } k < l\}, \quad (6)$$

and

$$\sigma = \bigcup_{k \in M} \sigma_k \quad (7)$$

A given disjunctive graph with a specific VAP solution defines a new disjunctive graph. Consequently, we could associate a disjunctive graph, $G_\psi = (N, A \cup \sigma, \bigcup_{m,k} \overline{E}_m^k)$ with each VAP solution ψ . Denote $A \cup \sigma$ as A_ψ , $\bigcup_{m,k} \overline{E}_m^k$ as E_ψ . We can then associate the disjunctive graph $G_\psi (N, A_\psi, E_\psi)$ with each solution ψ . Each disjunctive graph G_ψ defines a job-shop scheduling subproblem (P_ψ) for weighted tardiness as follows:

$$(P_\psi): \text{Minimize } \sum_{j \in 0} w_j (t_j + P_j - dd_j)^+ \quad (8)$$

s.t (Balas, 1979)

$$t_j - t_i \geq P_i \quad (i,j) \in A_\psi \quad (9)$$

$$t_j - t_i \geq P_i \vee t_i - t_j \geq P_j \quad (i,j) \in E_\psi \quad (10)$$

$$t_i \geq 0 \quad i \in N \quad (11)$$

(P_ψ) can be viewed as a variation of the original JSP with some additional precedence constraints (i.e., $A \subseteq A_\psi$).

As a distinct property of the above decomposition, an assignment decision is followed by a detail scheduling decision. The basic idea is to localize the sequencing decisions via a well-informed, higher level, operations assignment decision. For planning and control purpose, this specific property offers important planning stability, since much flexibility is retained for the schedule to handle shop dynamics.

3. Problem Analysis

3.1 Lower Bounds

Each solution of VAP(ψ) can be associated with a partially solved job-shop scheduling problem (P_ψ). The lower and upper bounds on problem P_ψ provide a performance index for each VAP solution, ψ . Suppose an operation assignment rather than a schedule is released at the beginning of a planning horizon, then the above index provides a pessimistic as well as optimistic estimation of the scheduling performance.

We have studied several lower bounds. A trivial lower bound can be computed by first dropping the unresolved disjunctive arcs in graph $G_\psi(N, A_\psi, E_\psi)$, then compute the ready time for each operation i by calculating the longest path from the source o to i . Denote this operation ready time $LB1(r_k)$, thus, we have the following relationship between each pair of arcs in A_ψ :

$$LB1(r_i) = \max_j(LB1(r_j) + p_j) \quad \forall (i,j) \in A_\psi. \quad (12)$$

A lower bound in weighted tardiness can be then computed as follows:

$$LB1(WT) = \sum_{k \in O} w_k(LB1(r_j) + p_k - dd_k)^+. \quad (13)$$

LB1 totally disregards the set E_ψ , thus it can be quite weak. Rather than ignoring E_ψ , we could improve the lower bound by using the processing times information of each subset. Define i as the set of operations assigned to an immediately preceding subset in the same machine of i . Thus the ready times of operation i is no less than the total processing time for i . That is, in general the earliest start time (EST) of subset k is no less than the EST of subset $k-1$ plus the sum of operation processing times in subset $k-1$. The corresponding lower bound on the operation ready time is as follows:

$$LB2(r_i) = \max \left\{ \min_{j \in i} \{LB2(r_j)\} + \sum_{j \in i} p_j, LB1(r_i) \right\}. \quad (14)$$

A new lower bound on weighted tardiness $LB2(WT)$ can be thus computed by (13) while replacing $LB1$ with $LB2$. In other words,

$$LB2(WT) = \sum_{k \in O} w_k(LB2(r_k) + p_k - dd_k)^+. \quad (15)$$

$LB2$ can be further tightened by considering following. Denote L_m the last subset of operations on machine m and C_m the earliest completion of machine m , thus,

$$C_m = \min_{i \in L_m} \{LB2(r_i)\} + \left\{ \sum_{i \in L_m} p_i \right\}. \quad (16)$$

Once each machine's earliest possible completion time is computed by C_m , we select the minimum possible weighted tardiness among the jobs which belong to machine m . A lower bound of weighted tardiness can be thus defined as follows:

$$LB3(WT) = \sum_{m \in M} \min_{j \in O_m} \{w_j \cdot (C_m - dd_j)^+\}. \quad (17)$$

where $O_m \subseteq O$ is the set of last operations on machine m . We then use the maximum of $LB2(WT)$ and $LB3(WT)$ as the lower bound $LB(WT)$. A more detail discussion of the lower bounds and their relationship to other scheduling lower bounds can be found in (Wu, Byeon and Storer, 1993).

3.2 Upper Bounds

As previous stated, associated with each VAP solution, is a more restricted disjunctive graph $G_\psi(N, A_\psi, E_\psi)$ and a scheduling problem P_ψ . An upper bound on P_ψ can be computed by completing the scheduling. We use the ATC dispatching heuristic of Vepsalainen and Morton (1987) to compute the upper bound of P_ψ .

4. Solution Methodology

As proposed by our formulation, to solve job shop scheduling problem we first solve a variant assignment problem (i.e., VAP) which puts operations in distinct subsets, we then solve a scheduling subproblem (P_ψ) over time by

generating the detailed schedule in a dynamic fashion. Thus, when solving VAP we assign operations to subsets so as to guarantee a certain level of scheduling performance. This can be achieved by using an assignment objective that reflects directly the scheduling objective. Unfortunately, this also makes the assignment objective extremely complex.

We first propose a heuristic which uses a linear approximation $\sum_i \sum_k c_{ik} X_{ik}$ of the true assignment objective.

We have developed a pricing structure which estimates the assignment costs c_{ik} based on the (scheduling) cost of assigning operation i to subset k . The VAP is then solved as a linear integer program. Each solution of this VAP represents a feasible assignment of operations to subsets. We iteratively adjust the linear assignment costs based on the lower or the upper bound computed for the corresponding scheduling problem ($P\psi$). In the rest of this paper, we will refer to the heuristics developed under this scheme as the Price Directed Heuristics, or PDH. Solving the integer program, obviously, can be quite time consuming. As an effort to speed up computation, we have developed a second set of heuristics which assign operations to subsets based on a priority index which estimates the impact of an assignment to detailed scheduling. Instead of a linear cost, each assignment is evaluated on the lower or the upper bound of its corresponding scheduling problem. We then iteratively adjust the parameters used in the priority index based on the computed cost. In the following discussion, we will call these heuristics Index Based Heuristics, or IBH. PDH and IBH are composed of distinct algorithmic components and many of them are shared by the two sets heuristics. In the following, we start with a detailed discussion of the PDH.

4.1 The Price Directed Heuristic(PDH)

(1) The Assignment Index

Given the job weight (w_j) and due-date (dd_j), we prioritize each operation corresponding to the weighted tardy objective. Since a due-date is give to each job, not to operations, we define an operation due-date as its latest finishing time (LFT) for meeting the job due-date. Similar approaches can be found in Baker and Kanet (1983). Based on the due-date dd_j of job j , an artificial due-date of operation i (LFT_i) can be defined as follows:

$$LFT_i = dd_j - \sum_k p_k \quad \forall (i,k) \in A. \quad (18)$$

Like most well-known dispatching rules (EDD, MINSLK etc.), more priority is assigned to job operation which has a heavier weight, tight due-date, or both. Normalized with job weight and operation due date (LFT), we define a priority index for each operation as follows:

$$\left(\frac{w_j - \min w}{\max w - \min w} \right) \cdot \left(1 - \frac{LFT_i - \min LFT}{\max LFT - \min LFT} \right) \quad (19)$$

The above index can be improved if we normalize it in the following fashion. Since for a job which has the smallest weight, all the operations of that job will have the same index (i.e., zero) regardless of their operation due-dates. Adding 1 to both terms in (19), we would normalize the index through the range of [1,4] instead of [0,1]. We define this as the priority index for each operation as follows:

$$\rho_i = \left(1 + \frac{w_j - \min w}{\max w - \min w} \right) \cdot \left(2 - \frac{LFT_i - \min LFT}{\max LFT - \min LFT} \right) \quad (20)$$

We use (20) as our primary priority index for assignment.

(2) Compute the Assignment Cost

As previously indicated, the linear cost c_{ik} in (1) is an estimation of the true assignment cost, or the corresponding (scheduling) cost of assigning operation i to subset k . In this section, we describe a linear, additive pricing structure which compute the price of having operation i assigned subset k based on its relative importance to the overall scheduling cost. A brief description of the pricing structure procedure is as follows:

Procedure PRICING STRUCTURE:

Step 0. Compute a priority index ρ_i for each operation $i \in N$. Sort the operations according to ρ_i in a nonincreasing order, and let S be the sorted list of operations without violating the precedence constraints, i.e., if $(i,j) \in A$, i precedes j in S .

For $i=1, \dots, |S|$

Given p subsets

For $k=1, \dots, p$

Step 1. Assign operation i to subset k , set $\alpha_k \leftarrow (\alpha_k - 1)$, and assign the remaining $S - \{i\}$ operations as follows:

For $j=1, \dots, p$

Assign the next α_j operation in $S - \{i\}$ to subset j with the following constraints:

(a) the predecessors of operation i can be only assigned to subsets $j \leq k$

(b) the successors of operation i can be only assigned to subsets $j \geq k$

end for

Step 2. Resolve disjunctive arcs $(i,j) \in E$ as follows:

if i is assigned to set m , j is assigned set n , and $m < n$, then fix direction $i \rightarrow j$.

Step 3. Compute a weighted-tardy lower bound for this graph, and set it as c_{ik} .

end for

Consider the following example: to compute the cost of assigning operation 7 to subset 3 ($C_{7,3}$) for a 10-machine, 5-subset problem, the procedure begins with assigning operation 7 to subset 3. This is followed by assigning the rest of the operation according to their order in $S - \{7\}$. Suppose $\alpha_1 = \alpha_2 = \dots = \alpha_5 = 20$. The predecessor of 7 may be assigned to either an earlier (subset 1 or 2) or the current subset (subset 3) depending on their priority index, but they can not be assigned to a later subset (e.g., subset 4 and 5). Based on the above assignment, a new disjunctive graph is defined. We then compute the cost $C_{7,3}$ as the weighted-tardy lower bound on the disjunctive graph.

The above pricing procedure computes the assignment cost C_{ik} for each operation i and subset k . Using this cost, we solve VAP as an integer linear programming ((1)-(5)) using LINDO. Based on the results from LINDO, we update the assignment cost in an iterative fashion, and VAP at each iteration.

4.2 The Index Based Heuristic (IBH)

We have implemented, in addition to the index in (20), a priority index originally proposed by Vepsalainen and Morton (1987). This priority index was originally developed for weighted tardy job shop scheduling problems. We use this index for operation assignment. The index can be briefly described as follows:

$$\frac{w_j}{p_j} \cdot \exp \left(- \left[\frac{dd_j - t - pi - \sum_{q \in E_i} (Wp + pq)}{kp} \right]^+ \right) \quad (21)$$

where, operation i is in job J and has the set of successors. The p in the denominator is the average processing time, and k is a *look-ahead* parameter (we set $k=3$). W_i is a leadtime estimator where we set $W_i = 2 \cdot p_i$. As in the PDH method, an assignment from this procedure leads to a solution to VAP, and a corresponding lower and upper bounds can be computed.

5. Experimental Results

We have implemented the PDH and IBH procedures in FORTRAN 77 on an IBM RS-6000 workstation. These sets of (weighted tardy) test problems, 10x10, 20x15 and 30x10, were generated from the makespan problems in Applegate and Cook (1991), kindly provided by William Cook. The 10x10 problems were generated by first obtaining the optimal makespan schedule using the Applegate and Cook's algorithm, then taking the completion times of each job in the schedule, subtract them by 100 and set these numbers as job due-dates. Job weights were generated by uniform random numbers in the range of [1,10]. The above scheme generates rather challenging tight due-date problems. The 20x15 and 30x10 problems were generated in a similar fashion except that a non-delay, instead of optimal, schedule were used for due-date assignment.

From a set of preliminary experiments, we found that the slack-based adjustment policy outperforms the other two policies and a step size ranging from 0.005 to 0.03 generally works well. We use this particular set-up throughout the computational testing. The computational results for PDH and IBH are reported in Table 1. As shown in the table, associated with each VAP solution is a lower and upper bounds (i.e., [LB UB]) to the corresponding scheduling problem (P_ψ). The upper bounds (UB) were generated by completing the schedule using the Vepsalainen and Morton's heuristic (VMH). As previously mentioned, if we view the VAP solution as a partial schedule then the lower and upper bounds provide a performance index of the assignment. The results for IBH were generated from 50-iteration runs while 500 iterations were used for IBH. This is due to the fact that PDH requires the solution of an integer program from LINDO which requires roughly 10 times the computation over IBH in each iteration.

As discussed earlier, both heuristics could iterate on either the upper bound (i.e., PDH(UB), IBH(UB)) or lower bounds (i.e., PDH(LB), IBH(LB)). To illustrate the effectiveness of the iterative search procedure itself, we have implemented an iterative version of Vepsalainen and Morton's heuristic. We use the slack-based adjustment policy in each iteration and 5000 iterations were used. Observed from columns VMH(0) and VMH(1) the iterative method appears to be very effective with an average improvement of 30%.

5.1 Robustness of the Partial Schedule

The results in Table 1 establish a rough comparison between the two VAP heuristics (PDH and IBH) and the more traditional scheduling heuristic VMH. Nevertheless, the results are rather difficult to interpret since the VAP heuristic generates only a partial schedule. A basic thesis of this paper is that this partial schedule provides global performance while offering local flexibility to handle shop disturbances. To verify this point we have conducted a set of Monte Carlo experiments which test robustness of the partial schedule under a wide range of shop disturbances. The results are then compared to the traditional static, and dynamic scheduling methods.

For each partial schedule (i.e., assignment) ψ generated in Table 1, we start with its corresponding disjunctive graph G_ψ and make the remaining scheduling decisions over time using a dynamic dispatching rule. To establish a fair comparison, we use Vepsalainen and Morton's ATC heuristic for the above dynamic dispatching, and we use the same heuristic to generate pure dynamic schedules where all scheduling decisions are made dynamically. Moreover, we use the iterative version of the ATC heuristic (i.e., VMH(I)) to generate static schedules. These scheduling procedures are then simulated under various levels of processing time variations. Specifically, we generate perturbed processing times p_i' as follows:

$$p_i' = p_i \pm \text{Exp}(\tau)$$

where $\tau = 5, 10, 15, 20, \dots, 60$. In the case where p_i' becomes negative, we set it to 1.

Due to space limitation, Table 2 summarizes only the simulation results for the case where $\tau = 15$. Each entry in the table represents the average total weighted tardiness of 100 simulation runs. The numbers in parenthesis represent percentage from best solution found. As can be seen from the table IBH methods in general, outperform the PDH. Furthermore, IBH(UB) is slightly better than IBH(LB).

We have conducted a more thorough set of simulation experiments using the IBH(UB) method as the assignment heuristic. In the simulation study, we further investigated the effect of different subset sizes. Specifically, for 10x10 problems, we limit ourselves to two subsets with three size configurations : [50,50], [25,75] and [75,25], where [x,y] represent x operations in first subset and y in the second. Similarly, for 20x15 and 30x10 problems we use [100,200], [150,150], and [200,100]. Quite clearly, the performance of static schedules deteriorates rapidly as the level of uncertainty increases. The VAP and the dynamic schedules, on the other hand, appear to be much more robust. In many cases, the VAP heuristics outperform both the static and the dynamic methods.

Table 1. Comparison of Heuristic Performance (WT) on test problems.

Problem	PDH(LB)		PDH(UB)		IBH(LB)		IBH(UB)		VMH(0)	VMH(I)
	[LB	UB]	[LB	UB]	[LB	UB]	[LB	UB]	UB	UB
<hr/>										
10 × 10	[LB	UB]	[LB	UB]	[LB	UB]	[LB	UB]	UB	UB
abz5	2886	8844	1792	8893	4077	8541	2334	9922	8532	7863
la16	117	4187	1870	4431	488	1781	519	3265	3688	1878
la20	2454	6044	1215	6225	1557	4019	1138	4613	5485	4533
mt10	875	7141	845	8260	1263	5172	1103	4510	6413	3486
orb6	447	3646	1171	5377	2052	4312	1760	5585	5572	3240
<hr/>										
30 × 10										
la31	2256	22581	1482	20815	922	7583	5969	7814	6558	4607
la33	4528	18868	4528	21120	2671	6143	4528	6645	5949	2708
la34	1743	16471	5590	17407	2144	16951	5376	13714	8780	6058
<hr/>										
20 × 15										
abz7	1114	10877	994	12463	1317	6711	1896	6031	6251	3677
abz8	2938	14973	3172	15399	1662	9842	3286	10939	9706	7297
abz9	179	10678	2672	11945	977	6848	2731	8314	8958	6918

PDH(UB):Price Directed Heuristic with iterative search based on Upper Bound.

PDH(LB):Price Directed Heuristic with iterative search based on Lower Bound.

IBH(UB):Index Based Heuristic with iterative search based on Upper Bound.

IBH(LB):Index Based Heuristic with iterative search based on Lower Bound.

VMH(0): Morton's ATC-Priority heuristic without iterative search.

VMH(I): Morton's ATC-Priority heuristic with iterative search.

Table 2. Simulation Results for $\mathcal{T} = 15$.

Problem	PDH(LB)	PDH(UB)	IBH(LB)	IBH(UB)
10 × 10				
abz5	10414	11823(.12)	10595(.02)	12878(.19)
la16	7386(.43)	6515(.36)	4414(.05)	4185
la20	8228(.36)	7556(.30)	6019(.12)	5287
mt10	10252(.19)	12650(.34)	8331	8715(.04)
orb6	6944	7326(.05)	8554(.19)	7645(.09)
30 × 10				
la31	28980(.35)	34019(.44)	18917	20789(.09)
la33	30542(.47)	30959(.48)	18333(.13)	16036
la34	28574(.20)	36573(.37)	28183(.19)	22878
20 × 15				
abz7	28367(.33)	31710(.40)	22044(.14)	18881
abz8	29233(.10)	30690(.14)	28228(.07)	26243
abz9	26462(.17)	23804(.08)	21986	22139(.01)
Ave %	.25	.29	.08	.04

PDH(UB):Price Directed Heuristic with iterative search based on Upper Bound.

PDH(LB):Price Directed Heuristic with iterative search based on Lower Bound.

IBH(UB):Index Based Heuristic with iterative search based on Upper Bound.

IBH(LB):Index Based Heuristic with iterative search based on Lower Bound.

6. Conclusions

This research describes a new decomposition method for the job shop scheduling problem using a variant of the classical assignment formulation. There are some characteristics of this approach which differentiate it from existing ideas dealing with JSP. First of all, using a graph theoretical approach, the proposed decomposition method offers computational efficiency. A machine scheduling and sequencing problem can be partitioned into a number of smaller problems so that less computational effort is required. Since many JSPs are too large to be solved economically by existing algorithms, the decomposition approach provides an obvious improvement. Secondly, for on-line control

purposes, decomposition provides more flexible and more robust schedules in situations involving shop disturbances.

References

- ADAMS, J., E. BALAS and D. ZAWACK, "The Shifting Bottleneck Procedure for Job Shop Scheduling," *Management Science*, Vol. 34, No. 3, 1988, pp. 391-401.
- APPLEGATE, D. and W. COOK, "A Computational Study of the Job-Shop scheduling Problem," *ORSA Journal on Computing*, Vol. 3, 1991, pp. 149-156.
- BAKER, K.R., "Sequencing Rules and Due-Date Assignments in a Job Shop," *Management Science*, Vol. 30, No. 9, Sept. 1984, pp. 1093-1104.
- BAKER, K.R. and J.J. KANET, "Job Shop Scheduling with Modified Due Dates," *Journal of Operations Management*, Vol. 4, No. 1, 1983, pp. 11-22.
- BALAS, E., "Disjunctive Programming," *Annals of Discrete Mathematics*, 5, 1979, pp. 3-51.
- BEAN, J.C., J.R. BIRGE, J. MITTENTHAL and C.E. NOON, "Matchup Scheduling with Multiple Resources, Release Dates and Disruptions," *Operations Research*, Vol. 39, No. 3, 1991, pp. 470-483.
- LAWLER, E.L., "A 'Pseudopolynomial' Algorithm for Sequencing Jobs to Minimize Total Tardiness," *Annals of Discrete Mathematics* 1, 1977, pp. 331-342.
- LAWRENCE, S.R. and T.E. MORTON, "Resource Constrained Multi-Project Scheduling with Tardy Costs: Comparing Myopic, Bottleneck, and Resource pricing Heuristics," *European Journal of Operational Research*, 64, 1993, pp. 168-187.
- MORTON, T.E., S. LAWRENCE, S. RAJAGOPALAN and S. KEKRE, "SCHED-STAR: A Price-Based Shop Scheduling Module", *Journal of Manufacturing and Operations Management*, 1/2, 1988, pp. 131-181.
- PINEDO, M. and M. Singer, "Shifting Bottleneck Heuristic for Minimizing the Total Weighted Tardiness in a Job Shop", *Naval Research Logistics*, Vol. 46, No. 1, pp. 1-17, 1999.
- POTTS, C.N. and L.N. VAN WASSENHOVE, "A Decomposition Algorithm for the Single Machine Total Tardiness Problem", *Operations Research Letters*, Vol. 1, No. 5, 1982, pp. 177-181.
- PRITSKER, A.A.B. and L.S. WATTERS, "A Zero-One Programming Approach to Scheduling with Limited Resources," *The RAND Corporation*, RM-5561-PR, January 1968.
- Ross, G.T. and R.M. SOLAND, "A Branch and Bound Algorithm for the Generalized Assignment Problem," *Mathematical Programming*, Vol. 8, 1975, pp. 91-103.
- ROUNDY, R., W. MAXWELL, Y. HERER, S. TAYUR and A. GETZLER, "A Price-Directed Approach to Real Time Scheduling of Production Operations", *IIE Transaction*, Vol. 23, No. 2, 1991, pp. 149-160.
- SIDNEY, J.B., "Decomposition Algorithms for Single Machine Sequencing with Precedence Relations and Deferral Costs", *Operations Research*, Vol. 23, No. 2, 1975, pp. 283-298.
- ULLMAN, J.D., "NP-Complete Scheduling Problems", *Journal of Computers and Systems Science*, 10, 1975, pp. 384-393.
- VEPSALAINEN, A.p. and T.E. MORTON, "Priority Rules for Job Shops with Weighted Tardy Costs," *Management Science*, Vol. 33, No. 8, 1987, pp. 95-103.
- WU, S.D., E. BYEON and R.H. STORER, "A Graph-Theoretic Decomposition of Job Shop Scheduling Problems to Achieve Scheduling Robustness", *Operations Research*, Vol. 47, No. 1, pp. 113-124, 1999.