Sādhanā, Vol. 22, Part 2, April 1997, pp. 241–256. © Printed in India.

An optimization-based algorithm for job shop scheduling

JIHUA WANG, PETER B LUH, XING ZHAO and JINLIN WANG

Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06269-2157, USA
e-mail: Luh@brc.uconn.edu

Abstract. Scheduling is a key factor for manufacturing productivity. Effective scheduling can improve on-time delivery, reduce inventory, cut lead times, and improve the utilization of bottleneck resources. Because of the combinatorial nature of scheduling problems, it is often difficult to find optimal schedules, especially within a limited amount of computation time. Production schedules therefore are usually generated by using heuristics in practice. However, it is very difficult to evaluate the quality of these schedules, and the consistency of performance may also be an issue.

In this paper, near-optimal solution methodologies for job shop scheduling are examined. The problem is formulated as integer optimization with a "separable" structure. The requirement of on-time delivery and low work-in-process inventory is modelled as a goal to minimize a weighted part tardiness and earliness penalty function. Lagrangian relaxation is used to decompose the problem into individual part subproblems with intuitive appeal. By iteratively solving these subproblems and updating the Lagrangian multipliers at the high level, near-optimal schedules are obtained with a lower bound provided as a byproduct. This paper reviews a few selected methods for solving subproblems and for updating multipliers. Based on the insights obtained, a new algorithm is presented that combines backward dynamic programming for solving low level subproblems and interleaved conjugate gradient method for solving the high level problem. The new method significantly improves algorithm convergence and solution quality. Numerical testing shows that the method is practical for job shop scheduling in industries.

Keywords. Scheduling; Lagrangian relaxation; dynamic programming.

1. Introduction

Scheduling is a key factor for manufacturing productivity. Effective scheduling can improve on-time delivery, reduce inventory, cut lead times, and improve the utilization of bottleneck resources. Because of the combinatorial nature of scheduling problems, it is often difficult to obtain optimal schedules, especially within a limited amount of computation

time. Production schedules therefore are usually generated by using heuristics in practice. However, it is very difficult to evaluate the quality of these schedules, and the consistency of performance may also be an issue. A logical strategy is thus to pursue methods that can consistently generate good schedules with quantifiable quality in a computationally efficient manner.

This paper examines the practical scheduling of job shops, a typical environment for the manufacture of low-volume and high-variety parts. In a job shop, parts with various due dates and priorities are to be processed on various types of machines. Job shop scheduling is to select the machines and beginning times for individual operations to achieve certain objective(s) with given machine capacities. In this paper, job shop scheduling is formulated as integer optimization with a "separable" structure. The requirement of on-time delivery and low work-in-process inventory is modelled as a goal to minimize a weighted part tardiness and earliness penalty function. Lagrangian relaxation (LR) is used to decompose the problem into individual part subproblems with intuitive appeal. By iteratively solving those subproblems and updating the Lagrangian multipliers at the high level, near-optimal schedules are obtained with a lower bound provided as a byproduct on the optimal cost. This paper reviews a few selected methods for solving subproblems and for updating multipliers. Based on the insights obtained, a new algorithm is presented that combines "backward" dynamic programming (BDP) for solving low level subproblems and interleaved conjugate gradient (ICG) method for solving the high level problem. The new method significantly improves algorithm convergence and solution quality. Numerical testing shows that the method is practical for job shop scheduling in industries.

1.1 *Literature review*

Given the economic and logistical importance of the scheduling problem, many of the early efforts centred on obtaining optimal schedules. Two prominent optimization methods are the branch and bound method (Fisher 1973) and dynamic programming (e.g., Pinedo 1995). It was discovered that the generation of optimal schedules often requires excessive computation time regardless the methodology. Furthermore, job shop scheduling is among the hardest combinatorial optimization problems and is NP-complete (Garey & Johnson 1979). Production schedules therefore are usually generated by experienced shop-floor personnel using simple dispatching rules in practice. Many heuristic methods have been presented and implemented based on due dates, criticality of operations, operation processing times, and machine utilization (e.g., Blackstone *et al* 1982). Many artificial intelligence (AI) approaches also use heuristics for scheduling (e.g., Kuziak 1990). These heuristics-based approaches usually generate feasible schedules quickly but it is very difficult to evaluate the quality of the schedules. Also, most heuristics do not provide for iterative improvement of the schedules.

Attempts to bridge the gap between heuristic and optimization approaches have also been undertaken (Adam *et al* 1988; Luh & Hoitomt 1993; Ventura & Weng 1995). In Adams *et al* (1988), for example, a heuristic for job shop scheduling was developed based upon optimally solving single machine sequencing problems. A criterion for measuring machine busyness was developed, and the job sequence for the busiest machine (the bottleneck) was first developed. The job sequence for the next busiest machine was then determined,

and the solution was fed back into the previously solved machine problem by a "local reoptimization." However, schedule evaluation could only be achieved through "selective enumeration." Also, each operation has to be pre-assigned to a specific machine before scheduling, though the operation may be processed on different machines or different types of machines.

In Luh & Hoitomt (1993), a Lagrangian relaxation framework was established for manufacturing system scheduling problems, and a practical method was provided. In the method, both machine capacity and operation precedence constraints are relaxed by using Lagrange multipliers, and operation-level subproblems are formed and solved by enumeration. The multipliers are then updated at the high level by using a subgradient method. An improved version of the method considering bills of materials, with a modified subgradient method at the high level was presented in Czerwinski & Luh (1994).

Much progress has been made on the scope and performance of the LR-based methodology. A combined LR and heuristic method was developed for job shop scheduling with group-dependent setups and finite buffers in Luh *et al* (1995). The scheduling of batch machines with setup requirements was addressed in Luh *et al* (1997b). A "forward" dynamic programming (FDP) algorithm was embedded within the LR framework for job shop scheduling in Chen *et al* (1995). In the method, only machine capacity constraints are relaxed, and part level subproblems are formed and solved by using the FDP. By doing this, the solution oscillation difficulties as reported in Czerwinski & Luh (1994) are alleviated. Also, by relaxing less constraints, the dual cost should be a tighter lower bound. In Luh *et al* (1997), an LR-based method was developed for job shop scheduling with uncertain parts. A "backward" dynamic programming (BDP) was developed to solve the part subproblems with random part parameters.

For high level algorithms, the slow convergence of subgradient methods was analysed, and the facet ascending algorithm (FAA) was presented to improve the convergence in Tomastik & Luh (1993). The reduced-complexity bundle method (RCBM) was developed in Tomastik & Luh (1996) which significantly reduces the computation complexity but maintains the convergence of the conventional bundle methods. An interleaved subgradient method (ISG) was developed in Kaskavelis & Caramanis (1995) to improve the efficiency of the LR-based method. A review of those methods will be presented in § 3.3.

1.2 *Overview of the paper*

In § 2, an integer optimization formulation with a "separable" structure for job shop scheduling is presented. In § 3, the problem is decomposed into individual part subproblems by relaxing machine capacity constraints following the approach in Chen *et al* (1995). A few selected methods for solving subproblems and for updating multipliers are reviewed. Based on the insights obtained, a new algorithm is presented that combines BDP for solving low level subproblems and an interleaved conjugate gradient (ICG) method for solving the high level problem. In § 4, numerical results show that the new method outperforms a previous LR/SG method in convergence. Numerical testing for practical data sets shows that the method can generate high quality schedules in a timely fashion.

2. Problem formulation

In a job shop, machines may have different processing capabilities. Machines with the same processing capability are grouped as a "machine type," and all the machine type form a set denoted as H. The total number of machine types is thus $|H|$. There are I parts with various due dates D_i to be scheduled over a discretized time horizon K. Part i ($i = 0, 1, \ldots, I-1$) consists of nonpreemptive J_i serial operations with operation j ($j = 0, 1, \ldots, J_i - 1$) of part i denoted by (i, j). An operation may start only after its preceding operation has been completed, and requires a machine belonging to a given set of eligible machine types H_{ij} for a specified duration of time. Without loss of generality, it is assumed that operations of part i are performed in the ascending order of operation index j.

The time horizon consists of K time units, indexed by k ($k = 0, 1, \ldots, K-1$). Each operation beginning time is defined as the beginning of the corresponding time unit, and each completion time the end of the time unit. The variables used in the problem formulation are listed below.

δ_{ijhk}: 0-1 operation variable which is one if operation (i, j) is performed on machine type h at time k, and zero other.
β_i: Part earliness weight.
b_{ij}: Beginning time of operation (i, j).
c_{ij}: Completion time of operation (i, j).
D_i: Due date of part i.
E_i: Earliness of part i, defined as $E_i = \max[0, S_i - b_{i0}]$.
h: Machine type variable, $h \in H$.
H_{ij}: Set of machine types capable of performing operation (i, j).
J: Objective function to be minimized.
k: Time index ($k = 0, 1, \ldots, K-1$).
M_{hk}: Capacity of machine type h at time k.
P_{ijh}: Processing time of operation (i, j) on machine type $h \in H_{ij}$.
S_i: Desired raw material release time for part i.
T_i: Tardiness of part i, defined as $T_i = \max[0, c_{i,J_i-1} - D_i]$.
W_i: Part tardiness weight.

Assuming that the set of machine types, the number of parts, part due dates and weights, operation processing time and time horizon are given, the constraints and objective function are explained below.

2.1 *Machine capacity constraints*

In the literature (e.g., Baker 1974; Adams 1988), the limited machine capacity is often modelled by "disjunctive constraints." Given a pair of operations (denoted as A and B) to be performed on a particular machine, the disjunctive constraints state that *either* B starts after the completion of A *or* A starts after the completion of B. As a result, the number of disjunctive constraints increase drastically with the number of operations. Also, it is

required that each operation must be pre-assigned to specific machines, though an operation may be processed on different machines or different types of machines.

By defining a set of 0-1 operation variables δ_{ijhk} to represent the processing status of each operation, the following machine capacity constraints are formed. The constraints state that the total number of operations being performed (active) on machine type h must be less than or equal to the capacity (M_{hk}) of machine type h at any time unit k, i.e.,

$$\sum_{i=0}^{I-1} \sum_{j=0}^{J_i-1} \delta_{ijhk} \leq M_{hk}, \quad h \in H; \quad k = 0, \ldots, K-1, \tag{1}$$

where the 0-1 operation variable δ_{ijhk} is defined by

$$\delta_{ijhk} = \begin{cases} 1, & \text{if } b_{ij} \leq k \leq c_{ij}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The number of machine capacity constraints equals the number of machine types times the time horizon K. Although the number of 0-1 operation variables is huge, these variables are determined once the machine types and beginning times of the operations are specified. They thus are not independent decision variables, and do not cause any complexity difficulty.

2.2 *Operation precedence constraints*

The operation precedence constraints are represented by the following "conjunctive constraints." These constraints state that an operation cannot be started until its preceding operation is finished, i.e.,

$$c_{i,j-1} + 1 \leq b_{ij}, \quad i = 0, 1, \ldots, I-1; \quad j = 1, 2, \ldots, J_i - 1. \tag{3}$$

Since operation $(i, j-1)$ is completed at the end of time unit $c_{i,j-1}$, and operation (i, j) starts at the beginning of time unit b_{ij}, the term "1" is required in (3). For the same reason, the term "1" also appears in the following processing time requirements.

2.3 *Processing time requirements*

The processing time requirements state that each operation must be assigned the required amount of time for processing on the selected machine type h, i.e.,

$$c_{ij} = b_{ij} + P_{ijh} - 1, \quad i = 0, 1, \ldots, I-1; \quad j = 0, 1, \ldots, J_i - 1; \quad h \in H_{ij}. \tag{4}$$

With processing times specified, operation completion times c_{ij} can be eliminated from the problem formulation. For notational convenience, they still appear in later derivation.

2.4 *Objective function*

Various objective functions such as makespan have been used in the literature. Research into practical scheduling, however, shows that the tardiness objective is likely to be more

useful than, say, makespan criteria (Blackstone *et al* 1982). In addition, the additivity of the tardiness objective function facilitates the decomposition approach.

Besides the on-time delivery, working-in-process (WIP) inventory is another major concern in practice. To reduce WIP inventory, a desired raw material release time S_i for each part is derived based on part due date and total processing time of the part (sum of operation processing times of the part). An earliness term for each part is added to the tardiness objective function, representing the penalty for releasing raw material too early. The requirement for on-time delivery and low WIP inventory is thus modelled as a goal to minimize the weighted part tardiness and earliness penalties, i.e.,

$$\mathbf{J} \equiv \sum_{i=0}^{I-1} (W_i T_i^2 + \beta_i E_i^2). \tag{5}$$

The square on tardiness reflect the fact that a part becomes more critical with each time unit after passing its due date. Similarly, square is applied to each earliness penalty term. The objective function accounts for the priorities of the parts, the importance of meeting due dates and desired release times.

The overall problem therefore is to minimize the part tardiness and earliness penalty function, subject to the above machine capacity and operation precedence constraints, i.e.,

$$\min_{\{b_{ij}, h_{ij}\}} \mathbf{J}, \text{ with } \mathbf{J} \equiv \sum_{i=0}^{I-1} (W_i T_i^2 + \beta_i E_i^2), \tag{6}$$

subject to

$$\sum_{i=0}^{I-1} \sum_{j=0}^{J_i-1} \delta_{ijhk} \leq M_{hk}, \quad h \in H; \quad k = 0, \ldots, K-1, \tag{7}$$

$$c_{i,j-1} + 1 \leq b_{ij}, \quad i = 0, 1, \ldots, I-1; \quad j = 1, 2, \ldots, J_i - 1. \tag{8}$$

The decision variables are the operation beginning times b_{ij} and the machine types h_{ij} for individual operations. Once b_{ij} and h_{ij} are selected, $\{c_{ij}\}$, $\{T_i\}$, $\{E_i\}$, and $\{\delta_{ijhk}\}$ can be easily derived.

3. Solution methodology

3.1 *Lagrangian relaxation*

Lagrangian relaxation (LR) is a mathematical programming technique for performing constrained optimization. Similar to pricing concept of a market economy, the Lagrangian relaxation method replaces "hard" coupling constraints (e.g., machine capacity constraints) by the payment of certain "prices" (i.e., Lagrange multipliers) for the use of machines at individual time units. The original NP-hard problem can thus be decomposed into many smaller and easier subproblems. The solutions of individual subproblems, when put together, may not constitute a feasible schedule since coupling constraints have been relaxed by the multipliers. These prices or multipliers are thus iteratively adjusted based on the degree of constraint violations following again the market economy mechanism.

Subproblems are then re-solved based on the new set of multipliers. In mathematical terms, a "dual function" is maximized in this multiplier updating process, and values of the dual function serve as lower bounds to the optimal feasible cost. At the termination of this multiplier updating process, simple heuristics are used to adjust subproblem solutions to provide a feasible schedule satisfying all constraints. Heuristics can also be run after each optimization iteration to check convergence or to provide candidate feasible schedules. Optimization and heuristics thus operate in a synergistic fashion to generate effective schedules. The quality of the schedule can also be quantitatively evaluated by comparing its cost to the largest lower bound provided by the dual function.

By using Lagrange multipliers π_{hk} to relax machine capacity constraints, the following relaxed problem is obtained.

Relaxed problem

$$\min_{\{b_{ij},h_{ij}\}} L, \text{ with } L \equiv \sum_i (W_i T_i^2 + \beta_i E_i^2) + \sum_{i,j} \sum_{k=b_{ij}}^{c_{ij}} \pi_{h_{ij}k} - \sum_{h,k} M_{hk}\pi_{hk}, \tag{9}$$

subject to the operation precedence constraints (3).

In deriving (9), the fact $\sum_{h\in H} \sum_{k=0}^{K-1} \pi_{hk}\delta_{ijhk} = \sum_{k=b_{ij}}^{c_{ij}} \pi_{h_{ij}k}$ is used, and the relaxed problem has b_{ij} and h_{ij} as its decision variables. After regrouping terms related to individual parts, the relaxed problem can be decomposed into the following part subproblems.

Part subproblems

$$\min_{\{b_{ij},h_{ij}\}} L_i, \text{ with } L_i \equiv W_i T_i^2 + \beta_i E_i^2 + \sum_{j=0}^{J_i-1} \sum_{k=b_{ij}}^{c_{ij}} \pi_{h_{ij}k}, \tag{10}$$

subject to the corresponding operation precedence constraints for part i.

From (10), a part subproblem reflects the needs to balance tardiness penalty, earliness penalty, and machine utilization costs. This part subproblem can be viewed as a multi-stage optimization problem with each stage corresponding to an operation. Although solving the original problem by using dynamic programming (DP) is impractical, the decomposed part subproblem is not NP-hard, and can be efficiently solved by using DP as will be presented in § 3.2.

Let L_i^* denote the minimal subproblem cost of part i with given multipliers, the high level Lagrangian dual problem is then obtained as below.

Dual problem

$$\max_{\{\pi_{hk}\}} D, \text{ with } D \equiv \sum_i L_i^* - \sum_{h,k} M_{hk}\pi_{hk}. \tag{11}$$

The Lagrangian dual function D is concave (Bertsekas 1995), and piece-wise linear, and consists of many "facets" (Tomastik & Luh 1993). We next present the resolution of part subproblems, followed by the updating of Lagrange multipliers.

3.2 *Dynamic programming*

The forward dynamic programming (FDP) algorithm presented by Chen *et al* (1995) can be used to solve a part subproblem in (10). It starts with the first operation of the part, and precedes to the last operation. In this paper, a backward dynamic programming (BDP) is developed with the goal to be further extended to handle uncertainties (e.g., uncertain arrival times, processing times, due dates etc., see Luh *et al* 1997). The BDP algorithm starts with the last stage, and compute the costs of the last operation $(i, J_i - 1)$ for all possible b_{i,J_i-1} and h_{i,J_i-1}:

$$V_{i,J_i-1}(b_{i,J_i-1}, h_{i,J_i-1}) = W_i T_i^2 + \sum_{k=b_{i,J_i-1}}^{c_{i,J_i-1}} \pi_{k h_{i,J_i-1}}. \tag{12}$$

For other operations (i, j), the cumulative costs are obtained by recursively solving the following DP equation subject to operation precedence constraints (3):

$$\begin{aligned} V_{ij}(b_{ij}, h_{ij}) &= \min_{b_{i,j+1}, h_{i,j+1}} \left\{ \Delta_{ij}\beta_i E_i^2 + \sum_{k=b_{ij}}^{c_{ij}} \pi_{h_{ij}k} + V_{i,j+1}(b_{i,j+1}, h_{i,j+1}) \right\} \\ &= \Delta_{ij}\beta_i E_i^2 + \sum_{k=b_{ij}}^{c_{ij}} \pi_{h_{ij}k} + \min_{b_{i,j+1}, h_{i,j+1}} V_{i,j+1}(b_{i,j+1}, h_{i,j+1}). \end{aligned} \tag{13}$$

In the above, Δ_{ij} is 1 if (i, j) is the first operation ($j = 0$) and 0 otherwise. The function $V_{ij}(b_{ij}, h_{ij})$ is the cumulative cost for all operations succeeding and including (i, j), and $\Delta_{ij}\beta_i E_i^2 + \sum_{k=b_{ij}}^{c_{ij}} \pi_{h_{ij}k}$ are the "stage-wise" costs. The algorithm starts from the last stage and moves backwards till the first stage is reached. The optimal subproblem cost L_i^* is then obtained as the minimal cumulative cost at the first stage. Finally, the optimal beginning times b_{ij} and machine types selected h_{ij} for operations can be obtained by tracing forwards the stages. Similar to FDP, the computation complexity of the above BDP algorithm is $O(K \sum_{j=1}^{J_i} |H_{ij}|)$ (Luh *et al* 1997).

3.3 *Solving dual problem*

3.3a *Subgradient methods*: As mentioned above, the Lagrangian dual function is concave and piece-wise linear. Existing methods for optimizing the dual function fall roughly into three classes: subgradient, cutting plane, and bundle methods. Of these, subgradient methods are commonly used to update the Lagrange multipliers (i.e., to maximize the dual function) because of their simplicity, the speed for computing a direction, and the global convergence property. With the subproblem solutions for given multipliers π_{hk}, the subgradient g of the dual function D is calculated by

$$g_{hk} = \sum_{i=0}^{I-1} \sum_{j=0}^{J_i-1} \delta_{ijhk} - M_{hk}, \quad h \in H; \quad k = 0, \ldots, K-1, \tag{14}$$

where g_{hk} is an element of the subgradient. In subgradient methods, multipliers are updated along the direction of the subgradient with the step size determined by

$$\alpha^n = \gamma \frac{D^* - D^n}{(g^n)^T g^n}, \qquad 0 < \gamma \leq 2 \tag{15}$$

where D^* is the optimal dual cost, and α^n, D^n and g^n are respectively the step size, dual cost and subgradient at iteration n. As shown in Tomastik & Luh (1993), subgradient methods often zigzag across a ridge (intersection of some facets) of the dual function. The slow convergence rate (less than linear) of the subgradient methods causes these methods to require many iterations to reach an optimum.

3.3b *Facet ascending algorithm*: By recognizing that the Lagrangian dual function is polyhedral concave, and is made up of many facets, the facet ascending algorithm (FAA) finds the intersection of adjacent facets (Tomastik & Luh 1993). A subgradient of one of the facets is then projected on the intersection to obtain an ascending direction, and a line search technique is used to determine how far to move along the direction. The FAA avoids the zigzagging behaviour of subgradient methods, and shows improved convergence. For large problems, an intersection is usually formed by many facets. Finding such an intersection is often difficult and requires many dual function evaluations which are "computationally expensive." Furthermore, the ridges are short, causing slow convergence.

3.3c *Bundle methods*: The bundle method (e.g., Hiriart-Urruty & Lemarechal 1993) has the fastest convergence rate among the three classes of methods. It accumulates and utilizes the subgradients of points within a neighbourhood of the current iterate to find an ϵ-ascent direction (along which the function value can increase at least by ϵ), or to detect within ϵ of the dual optimum (ϵ-optimal). Finding such a direction or detecting ϵ-optimal, however, requires solving a number of quadratic programming problems with considerable complexity. To reduce the complexity while maintaining the convergence of the bundle method, the reduced-complexity bundle method (RCBM) finds an ϵ-ascent direction by performing a projection of a subgradient onto an appropriate subspace formed by the subgradients in the bundle (Tomastik & Luh 1996). Along the ϵ-ascent direction, a line search technique is then used to determine the step size for updating multipliers. Similar to FAA, the large number of dual function evaluations required to accumulate the subgradients and to perform line search is very time consuming, and hinders the applicability of the RCBM to very large problems.

3.3d *Interleaved subgradient method*: The iterative resolution of the dual problem requires the dual function to be evaluated many times, and each function evaluation involves solving all the subproblems once (called one iteration). These dual function evaluations are extremely "expensive" for large problems. For example, it takes about 68% of total CPU time for a case with 82 parts and 14 machines. To efficiently utilize the expensive function

evaluations, an interleaved subgradient (ISG) method has been developed (Kaskavelis & Caramanis 1995). Instead of solving all subproblems before updating multipliers, the ISG method updates multipliers after solving each subproblem. At the high level, the multipliers are updated along the direction of the subgradient. Numerical results show that the ISG method converges much faster than a subgradient method, though algorithm convergence has not yet been established.

3.3e *Interleaved conjugate gradient method*: As mentioned earlier, the dual function is concave, piece-wise linear, and consists of many facets. Each possible solution of the relaxed problem corresponds to a facet. Because of the combinatorial nature of the original problem, the number of possible solutions of the relaxed problem and therefore the number of facets increases drastically as the problem size increases. The dual function thus approaches a smooth function. This "smoothness" of the dual function motivates the use of optimization methods for smooth functions.

Among the methods for optimizing smooth functions, conjugate gradient methods have attractive convergence properties and computation efficiency. The conjugate directions are generated by

$$d^n = g^n + \beta^n d^{n-1} \text{ with } d^0 = g^0, \quad \beta^n = \frac{(g^n)^T g^n}{(g^{n-1})^T g^{n-1}}, \quad n = 1, 2, \ldots. \tag{16}$$

where d^n and g^n are the conjugate direction and gradient at iteration n. The step size for updating multipliers is determined by performing a line search along the conjugate direction.

By incorporating the "interleave" concept with the conjugate gradient method, an interleaved conjugate gradient (ICG) method has been developed that utilizes the "smooth" property of the dual function and efficiency of the interleaved method for problems of large sizes (e.g., 1000 multipliers or more). In this paper, the ICG method is used to update the multipliers. The ICG algorithm is summarized as follows.

S0 Given the initial multipliers, solve all the part subproblems, and compute the dual cost and subgradient. Update multipliers along the direction of the subgradient. Set subproblem index $s = 1$.

S1 Solve subproblem s while keeping other subproblem solutions unchanged. Compute the "surrogate" dual cost and subgradient according to (11) and (14) with the latest available subproblem solutions.

S2 Compute conjugate direction by (16) with the surrogate subgradient, and update multipliers along the direction. Since only one subproblem is solved for a set of multipliers, line search cannot be used to determined the step size. The step size is therefore still computed according to (15) with D^* replaced by the lowest feasible cost obtained up to the current iteration.

S3 Increase s by one. If s is larger than the total number of subproblems, reset s to 1. Go to S1.

3.4 *Constructing feasible schedule*

The solutions to part subproblems, when put together, are generally associated with an infeasible schedule, i.e., capacity constraints might be violated at some time periods. A feasible schedule is constructed by using a list scheduling heuristic. In the list scheduling procedure, a list of immediately performable operations is created, and maintained in the ascending order of their beginning times from part subproblem solutions. Operations are then scheduled on the required machine types according to this list as machines become available. If the capacity constraint for a particular machine type is violated at time k, a greedy heuristic determines which operations should begin at that time and which ones are to be delayed by one time unit. The subsequent operations of those delayed ones are then delayed by one time unit if precedence constraints are violated. The process repeats until the last operation in the list.

The cost of the feasible schedule $\mathbf{J}$ is an upper bound on the optimal cost $\mathbf{J}^*$. The optimal dual value D^*, on the other hand, is a lower bound on $\mathbf{J}^*$. Since it is usually difficult to find $\mathbf{J}^*$ and D^*, the (relative) duality gap$(\mathbf{J} - D)/D$ is often used as a measure of the quality of the feasible schedule.

4. Numerical results

The new method that combines BDP and ICG within the LR framework has been implemented using the object-oriented programming language C++, and extensive testing has been performed. Four test cases are presented below. The first two cases are to demonstrate that the new LR/BDP/ICG method has better convergence than the previous LR/SG method with both machine capacity and operation precedence constraints relaxed and with a subgradient method at the high level. Case 2 also shows that the LR/BDP/ICG method generates a better schedule than a heuristic method that combines the "first come first serve (FCFS)" and "shortest processing time (SPT)" rules (called FCFS/SPT). The next two cases demonstrate that the LR/BDP/ICG method is applicable for solving scheduling problems of realistic sizes. In presenting the results for both LR/SG and LR/BDP/ICG methods, an iteration corresponds to solving all subproblems once.

The four cases are tested on a Sun Sparc 10 workstation. In the testing, all multipliers are initialized at zero. The time horizons are automatically generated based on machine availabilities and part processing requirements. The step size factor γ in the LR/BDP/ICG method is initialized to a specific value, and adaptively adjusted based on information obtained in the iterative process.

Case 1: This test case is to demonstrate that the LR/BDP/ICG method generates a tighter lower bound than the LR/SG method. There are two machines of different types and two parts. Part one has two operations with processing times 3 and 2, respectively, and part two also has two operations with processing times 1 and 4, respectively. The first operations of both parts require machine type 0, and the second operations require machine type 1. The due dates are zero. The tardiness penalty weights are 1, and there is no earliness penalty. The results are summarized in table 1.

Table 1. Testing results for case 1 ($\gamma = 0.5$).

Method	Iteration	Dual/**J**	Duality gap	CPU (s)
LR/SG	2000	44/52	18%	3.00
LR/BDP/ICG	17	52/52	0%	0.01

It can be seen that both methods generate optimal schedule with the cost $\mathbf{J} = 52$ which equals to the lower bound obtained by LR/BDP/ICG. The dual cost by using LR/BDP/ICG is thus optimal. The dual cost obtained by LR/SG (= 44) is also optimal for the corresponding dual function, as verified by using a LR/RCBM method. This test case therefore shows that by using the above LR/SG method, there exists an inherent gap (defined as the gap between the optimal dual cost and optimal feasible cost). It also shows that a tighter bound is obtained when the operation precedence constraints are not relaxed.

Case 2: This test case is to demonstrate that the LR/BDP/ICG method outperforms the LR/SG method and the FCFS/SPT method. In this case, there are three machine types with one machine each, and four parts with a total of twelve operations. For all the parts, the due date and weight are -1 and 5 respectively, and there is no part earliness penalty. Operation processing times and required machine types are listed in table 2a, and the time horizon is 30. Testing results are summarized in tables 2b and 2c.

Since all the weights are integer, the cost of a feasible schedule should also be an integer. With the lower bound 2374.7 and feasible cost 2375 obtained by LR/BDP/ICG, the schedule obtained must thus be optimal. It can also be seen from table 2b that the LR/BDP/ICG method significantly speeds up convergence.

This case is also tested by using the FCFS/SPT method. In the FCFS/SPT method, operations are performed according to the FCFS rule. When several operations come to a machine at the same time, the SPT rule is used to determine the sequence of the these operations. The cost of the resulting schedule is 2852, which is significantly higher than the optimal feasible cost (= 2375). The schedule obtained by FCFS/SPT as well as the optimal schedule is presented in table 2c.

The following two cases draw data from industries, and are tested by using the LR/BDP/ICG method. The purpose is to demonstrate that the method is applicable for solving practical scheduling problems. A few performance metrics as well as the feasible costs at some iterations are evaluated to measure the schedule quality. The metrics are defined below.

Makespan: the duration of time for processing all the parts.

Maximum work-in-process inventory: the maximum number of parts in processing at a time unit.

Average work-in-process inventory: the average number of parts in processing over the makespan.

Average lead time: the average elapse time between part beginning and completion times for all parts.

Table 2a. Input data for case 2 ($\gamma = 0.5$).

Part	(Processing time/Machine type)		
	Operation 0	Operation 1	Operation 2
Part 0	4/0	3/1	2/2
Part 1	1/1	4/0	4/2
Part 2	3/2	2/1	3/0
Part 3	3/1	3/2	1/0

Table 2b. Testing results for case 2.

Method	Iteration	Dual cost	J	Duality gap	CPU (s)
LR/SG	2000	2179	2375	9.0%	4.0
LR/BDP/ICG	100	2374.7	2375	0.1%	0.1

Table 2c. Feasible schedules for case 2.

(i, j)	h	LR/BDP/ICG		FCFS/SPT	
		b_{ij}	c_{ij}	b_{ij}	c_{ij}
(0, 0)	0	0	3	0	3
(0, 1)	1	4	6	6	8
(0, 2)	2	7	8	12	13
(1, 0)	1	3	3	0	0
(1, 1)	0	4	7	4	7
(1, 2)	2	9	12	8	11
(2, 0)	2	0	2	0	2
(2, 1)	1	7	8	4	5
(2, 2)	0	9	11	9	11
(3, 0)	1	0	2	1	3
(3, 1)	2	3	5	4	6
(3, 2)	0	8	8	8	8

Case 3: This case is to show that the method selects machine types for individual operations as well as their beginning times. In this case, there are eleven machine types with a total of 16 machines and 18 parts with various due dates and weights. A part may have up to 17 operations, and the total number of operations is 159. An operation may be performed on one of up to four different machine types. With a time horizon 1086, the testing results are summarized in table 3a, and the performance metrics in table 3b.

It can be seen that both the lower bound and feasible schedule keep on improving as the number of iterations increases. It is also shown in table 3b that the schedule obtained at iteration 400 has better performance than the schedule at iteration 1.

Case 4: This case is to demonstrate the capability of the LR/BDP/ICG method for scheduling problems with large sizes. In this case, there are eight machine types with a total of 14 machines. Each machine type has one or two identical machines. A total of 82 parts with various due dates and weights are scheduled over a time horizon of 2068. A part may have up to 17 operations, and the total number of operations for all parts is 752. An

Table 3a. Testing results for case 3 ($\gamma = 0.1$).

Iteration	Dual cost	J	Duality gap	CPU (s)
1	19339	86988	350%	0.7
200	37394	40715	8.9%	128
400	37456	40170	7.2%	261

Table 3b. Performance metrics.

Iteration	1	200	400
Makespan	782	544	546
Maximum work-in-process inventory	18	16	16
Average work-in-process inventory	8.16	8.20	8.09
Average lead time	354	247	245

operation can be performed on a specific machine type. Testing results are summarized in table 4a, and the performance metrics in table 4b.

The results in tables 4a and 4b show the iterative improvement of the LR/BDP/ICG method on dual costs, feasible costs and performance metrics.

From the results of cases 3 and 4, it can be seen that significant improvement on the schedule quality and lower bound is obtained in the first 200 iterations. The improvement slows down from iteration 200 to 400. It is thus not needed to run the algorithm for a long time to get high quality schedules.

As mentioned above, the computation complexity of the BDP algorithm for solving part subproblems is $O(K \sum_{j=1}^{J_i} |H_{ij}|)$. It is observed from testing that the resolution of subproblems takes most of the CPU time in LR/BDP/ICG. The complexity of the LR/BDP/ICG algorithm is thus dominated by BDP. As shown in table 5, the computation time for 200 iterations increases almost linearly with $K \sum_{j=1}^{J_i} |H_{ij}|$.

Lagrange multipliers reflect the "price" information for using machines. In day-to-day scheduling, the multipliers associated with the previous schedule can initialize the algorithm to generate a new schedule. Since the production in a job shop may not change much from day-to-day, the computation time for finding a good schedule will

Table 4a. Testing results for case 4 ($\gamma = 0.1$).

Iteration	Dual	J	Duality gap	CPU (s)
1	84.5	46487	54914%	3
200	28837	38843	34.7%	542
400	29921	37161	24.0%	1086

Table 4b. Performance metrics of feasible schedule.

Iteration	1	200	400
Makespan	1049	1050	1047
Maximum work-in-process inventory	42	30	32
Average work-in-process inventory	22.3	15.4	16.8
Average lead time	322	219	240

Table 5. Computation time analysis for cases 3 and 4.

Case	$\sum_{j=1}^{J_i} \lvert H_{ij} \rvert$	K	$K \sum_{j=1}^{J_i} \lvert H_{ij} \rvert$ ($\times 10^3$)	CPU (s)/200 iter.
3	269	1086	292	128
4	752 (269 × 2.8)	2068 (1086 × 1.9)	1555 (292 × **5.3**)	542 (128 × **4.2**)

decrease vastly (roughly by 2/3 according to testing experience) with initialization procedure.

5. Conclusions

In this paper, near-optimal solution methodologies for job shop scheduling are examined, and many insights are provided on a few selected methods for solving subproblems and for updating multipliers. A new algorithm is presented that combines backward dynamic programming for solving low level subproblems and interleaved conjugate gradient method for solving the high level problem. The new method significantly improves algorithm convergence and solution quality. Numerical testing for practical data sets shows that the LR/BDP/ICG method can generate high quality schedules in a timely fashion, and it is practical for job shop scheduling in industries.

This work was supported in part by the National Science Foundation under DMI-9500037, and the Advanced Technology Center for Precision Manufacturing, University of Connecticut. The authors would like to thank Mr. Ling Gou and Mr. Bin Jin of the University of Connecticut for their valuable help in the algorithm development.

References

Adams J, Balas E, Zawack D 1988 The shifting bottleneck procedure for job shop scheduling. *Manage. Sci.* 34: 391–401

Baker K 1974 *Introduction to sequencing and scheduling* (New York: Wiley)

Bertsekas D P 1995 *Nonlinear programming* (Belmont, MA: Athena Scientific)

Blackstone J H, Phillips D T, Hogg G L 1982 A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *Int. J. Product. Res.* 20: 27–45

Chen H, Chu C, Proth J M 1995 A more efficient Lagrangian relaxation approach to job-shop scheduling problems. *Proc. IEEE Int. Conf. on Robotics and Automation*, pp 496–501

Czerwinski C, Luh P B 1994 Scheduling parts with bills of materials using an improved Lagrangian relaxation technique. *IEEE Trans. Robotics Autom.* 10: 99–111

Fisher M L 1973 Optimal solution of scheduling problems using Lagrange multipliers. Part I. *Oper. Res.* 21: 1114–1127

Garey M R, Johnson D S 1979 *Computers and intractability* (San Francisco: WH Freeman)

Hiriart-Urruty J B, Lemarechal C 1993 *Convex analysis and minimization algorithms* (Berlin: Springer-Verlag) vols 1 & 2

Kaskavelis C A, Caramanis M C 1995 Efficient Lagrangian relaxation algorithms for real-life-size job-shop scheduling problems. Working Paper, Department of Manufacturing Engineering, Boston University; also personal communications

Kuziak A 1990 *Intelligent manufacturing systems* (Englewood Cliffs, NJ: Prentice-Hall)

Luh P B, Hoitomt D J 1993 Scheduling of manufacturing systems using the Lagrangian relaxation technique. *IEEE Trans. Autom. Control.* 38: 1066–1079

Luh P B, Gou L, Odahara T, Tsuji M, Yoneda K, Hasegawa T, Kyoya Y 1995 Job shop scheduling with group-dependent setups, finite buffers, and long time horizon. *Proceedings of the 34th Conference on Decision and Control*, New Orleans, LA, pp 4184–4189

Luh P B, Chen D, Thakur L S 1997a Modelling uncertainty in job shop scheduling. *Proc. of the First International Conference on Operations and Quantitative Management*, Jaipur, India, pp 490–497

Luh P B, Wang J H, Wang J L, Tomastik R N 1997b Near optimal scheduling of manufacturing systems with presence of batch machines and setup requirements. *Ann. CIRP* 46: (to appear)

Pinedo M 1995 *Scheduling – Theory, algorithms and systems* (Englewood Cliffs, NJ: Prentice Hall)

Tomastik R N, Luh P B 1993 The facet ascending algorithm for integer programming problems. *Proc. IEEE Conf. on Decision and Control*, San Antonio, Texas, pp 2880–2884

Tomastik R N, Luh P B 1996 A reduced-complexity bundle method for maximizing concave non-smooth functions. *Proc. of the 31st Conference on Decision and Control*, Kobe, Japan, pp 2114–2119

Ventura J A, Weng M X 1995 Minimizing single-machine completion time variance. *Manage. Sci.* 41: 1448–1455