Scheduling Job Shops with Batch Machines Using the Lagrangian Relaxation Technique

Jihua Wang and Peter B. Luh
The University of Connecticut
Dept. of Electrical & Systems Engineering
Storrs, CT 06269-2157, USA

Phone: (860) 486-4821 Fax: (860) 486-2447
Email: Jihua@brc.uconn.edu, Luh@brc.uconn.edu

Abstract

Scheduling is a key factor for manufacturing productivity. Effective scheduling can improve on-time delivery, reduce work-in-process inventory, cut lead time, and improve machine utilization. Motivated by the extensive use of batch machines in manufacturing industries, the scheduling of job shops with *batch machines* is studied in this paper. Unlike machines that can process one part at a time (called "standard machines" for simplicity), a batch machine is the one that can *simultaneously* process several parts with the same processing requirement as a *batch*, subject to the capacity of the batch machine. This simultaneous processing requires the "synchronization" (batch formation) of different parts for a batch operation. In view that a part may have to be processed by many standard and batch machines, multiple batch formation and batch splitting result in complicated coupling among the parts.

In this paper, batches are viewed as "virtual" facilities that host and "synchronize" parts for processing, and compete for batch machine capacity. A novel integer programming formulation with a "separable" structure is then obtained. To solve this problem, a Lagrangian relaxation approach is used to decompose the problem into part and batch subproblems that can be solved by using an efficient dynamic programming algorithm. The multipliers are updated at the high level by using an “interleaved conjugate gradient method,” and a heuristic is developed to obtain feasible schedules based on subproblem solutions. Numerical testing shows that the integrated consideration of batch formation and sequencing results in high quality schedules, and the algorithm is computationally efficient to solve practical problems.

Keywords: Scheduling, Job shop, Batch machine, Lagrangian relaxation

1. INTRODUCTION

Scheduling is a key factor for manufacturing productivity. Effective scheduling can improve on-time delivery of products, reduce work-in-process inventory, cut lead times, and improve machine utilization. In discrete manufacturing systems, a machine often processes only one part at a time (called "standard machine" for simplicity). There are also many "batch machines," where a batch machine is the one that can *simultaneously* process multiple parts with the same processing requirement as a batch, subject to the capacity of the batch machine. For example, a heat-treat oven can simultaneously process multiple parts with the same processing requirement (temperature, processing time etc.) in a batch. A 3-axle NC machine can perform the same operation of up to three parts simultaneously. The high utilization requirement and the long non-preemptive processing times place great efficiency demand on these machines. Motivated by the extensive use of batch machines in discrete manufacturing systems and the associated demand on high efficiency, this paper presents an optimization-based method for the scheduling of job shops with both standard and batch machines.

The scheduling of batch machines requires the grouping of parts into batches (batch formation) and the sequencing of batches. Parts with the same processing requirement belong to a "group," and parts from different groups cannot be processed in the same batch. Since a batch machine can simultaneously process multiple parts, it may be desirable to form a batch with as many parts as possible. However, parts assigned to a batch may not be available at the same time, and the parts available earlier have to wait for processing until all the parts become available. This waiting may cause significant delay of the parts available earlier, resulting in poor scheduling performance. The trade-off between utilization and due date performance has been observed to be a major difficulty for many factories (Zijm, 1995). To overcome the difficulty, it is required to consider batch formation and sequencing in an integrated fashion. However, in view that a part may have to be processed by many standard and batch machines, multiple batch formation and batch splitting result in complicated coupling among these parts.

Literature Review

In view of the difficulties in scheduling batch machines, the problem is often separated into batch formation and batch sequencing problems. In Ahmadi et al. (1992), the scheduling of a two- or three-

machine flow shop with one batch machine is considered as a two-stage batch formation and sequencing problem. Makespan and the sum of job completion times are used as the performance criteria, and a heuristic method is presented. A two-stage batch formation and batch sequencing algorithm for NC punch presses is developed in Lee et al. (1993). At the first stage, batches are formed to maximize the number of parts in batches and the tool magazine utilization. At the second stage, batches are sequenced to reduce the total setup time by using the "nearest neighbor" heuristic.

Methods for the integrated consideration of batch formation and sequencing have also been presented. The scheduling of parts belonging to multiple groups on a batch machine are considered in Uzsoy (1995). The objectives of makespan, maximum lateness and total weighted completion time are discussed. It is shown that when all parts are available simultaneously, keeping a batch as full as possible gives the optimal solution. Several heuristics are developed to minimize the maximum lateness for dynamic job arrivals. In Dessouky et al. (1996), a mixed integer nonlinear model with the objective to minimize the total tardiness is developed for the scheduling of batch chemical plants. A heuristic algorithm is developed to determine the sizes of batches and their schedule to satisfy the demand of outstanding orders.

Optimization-based approaches have also been reported. In Liao et al. (1993), a combined Lagrangian relaxation and linear network flow algorithm is developed for the scheduling of a flow line of semiconductor wafer fabrication with batch machines. Batch formation is to determine the batch size, subject to the capacity of the batch machine and flow balance equations. In Wang et al. (1994), the scheduling of a single batch machine is studied. A separable problem formulation accommodating both batch formation and batch sequencing is presented. The large number of constraints involved, however, limits its application to problems of small sizes.

Overview of the Paper

In this paper, the scheduling of a job shop with batch machines is studied with an integrated consideration of batch formation and sequencing. Because of the combinatorial nature of the problem, it is very difficult to obtain optimal schedules, especially within a limited amount of computation time. The goal of the study, therefore, is to obtain near-optimal schedules with quantifiable quality in a computationally efficient manner. To achieve this, a Lagrangian relaxation

based method is developed to decompose the problem into smaller subproblems that can be efficiently solved without encountering complexity difficulty. The key for the effective application of Lagrangian relaxation is to have a problem formulation with a “separable” structure – the objective function and all "coupling constraints" are additive in terms of basic decision variables. The synchronization, however, leads to the strong couplings among parts, making it difficult to have a separable formulation.

To overcome the difficulties, batches are viewed as "virtual" facilities that host and "synchronize" parts for processing, and compete for batch machine capacity. The hosting of parts and the competition for batch machines are delineated by two sets of linear constraints. A separable integer programming formulation with manageable numbers of variables and constraints is then obtained in Section 2. In Section 3, a solution methodology based on the synergistic combination of Lagrangian relaxation and heuristics is developed. Within the Lagrangian relaxation framework, dynamic programming (DP) is used to solve subproblems as in Chen, Chu and Proth (1995). The multipliers are updated at the high level by using the "interleaved conjugate gradient (ICG) method," and a heuristic is developed to generate feasible schedules based on subproblem solutions. The solution method has been implemented using the object-oriented programming language C++. Numerical testing presented in Section 4 shows that the integrated consideration of batch formation and sequencing results in high quality schedules, and the algorithm is computationally efficient to solve practical problems.

2. PROBLEM FORMULATION

The formulation to be presented is built on our previous work on job shop scheduling (Hoitomt et al., 1993) and single batch machine scheduling (Wang et al., 1994) while considering multiple batch machines and the interaction among standard and batch machines. A general description of the problem is provided in Subsection 2.1. The constraints and objective function are then presented in Subsections 2.2, 2.3 and 2.4. A list of symbols used is provided in Appendix A for easy reference.

2.1 General Description

According to processing capabilities, machines in the shop are grouped into a set of machine types, denoted by H, where a machine type $h \in H$ may consist of a few identical machines. The set for all batch machine types is H_B, and the set for all standard machine types is $H_{\overline{B}}$. The number of machines available for a machine type h, denoted as M_{hk}, may change over a discretized time horizon K with time index k ranging from 0 through K - 1. For a batch machine of type h, the number of parts in a batch is restricted by the batch volume V_h of machine type h (e.g., the volume of a heat treat oven or the maximum number of parts on an NC machine).

There are I parts to be scheduled. Part i (i = 0, 1, …, I - 1) has a desired release time $\overline{b}_i$ and a given due date D_i, and it consists of J_i non-preemptive operations. Operation j (j = 0, 1, …, J_i - 1) of part i, denoted as operation (i, j), requires a machine of type h belonging to a set of eligible machine types H_{ij} to process for P_{ijh} amount of time. Operations to be performed on standard machines are called "standard operations," and operations on batch machines are called "batch operations." All operations that can be processed on a batch machine type $h \in H_B$ belong to G_h groups, and each operation occupies the batch volume V_h with a size (or a volume) v_{ij}. Operations belonging to group g (g = 0, 1, …, G_h - 1) are processed in N_{hg} batches, each requiring P_{hg} amount of processing time, and $P_{hg} = P_{ijh}$ if operation (i, j) belongs to the group. The batches indexed by n (n = 0, 1, …, N_{hg} - 1) are assigned to individual machines, and batches on each machine are processed in the ascending order of n. The number of batches within a group is given, and is assumed to be large enough to hold all batch operations of the group. Batch n (n = 0, 1, …, N_{hg} - 1) of group g on machine type h is denoted as batch (h, g, n), and group g on machine type h denoted by (h, g).

In formulating the scheduling problem, batches are viewed as virtual facilities that host and synchronize operations on parts. The synchronization among operations is modeled as that between operations and the virtual facilities. To assign an operation to a batch machine type with beginning time k, a virtual facility (batch) should be available to host the operation, and the volume sum of the operations should not exceed the batch volume. Batches are also viewed as "virtual" operations that compete for batch machine capacity, and a batch machine can process at most one virtual operation at a time.

Different from the above batch operations, operations performed on standard machine types directly compete machine capacity, that can be modeled as in Hoitomt, Luh, and Pattipati (1993).

2.2 Modeling of Standard Machine Types

The *machine capacity constraints for standard machines* state that the number of operations assigned to a standard machine type h cannot exceed the number of type h machines available at time k, i.e.,

$$\sum_{i=0}^{I-1} \sum_{j \in \overline{B}_i} \delta_{ijhk} \le M_{hk}, \quad h \in H_{\overline{B}}, \; k = 0, 1, \ldots, K - 1, \tag{1}$$

where δ_{ijhk} is a 0-1 *operation in-processing variable*. It is 1 if operation (i, j) is performed on a machine of type h at time k, and 0 otherwise. Set $\overline{B}_i$ is the set of all the standard operations of part i. Although the number of operation in-processing variables δ_{ijhk} is huge, these variables are not independent decision variables. They are determined once the machine types for individual operations are selected, and operation beginning times are determined, i.e.,

$$\delta_{ijhk} = \begin{cases} 1 & \text{if operation (i, j) is assigned to machine type h and } b_{ij} \le k \le c_{ij}; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

2.3 Operation Precedence Constraints and Processing Time Requirements

The *operation precedence constraints* state that an operation cannot start until its preceding operation is finished, i.e.,

$$c_{ij} + S_{ij} - 1 \le b_{i,j+1}, \; \forall (i, j), \tag{7}$$

where c_{ij} is the completion time of operation (i, j), and $b_{i,j+1}$ the beginning time of operation (i, j+1). The parameter S_{ij} is the required "timeout" between operations (i, j) and (i, j+1), representing the

processes not explicitly modeled in the problem formulation (e.g., the transportation time required in-between the two operations).

The *operation processing time requirements* state that each operation must be assigned the required amount of processing time P_{ijh} on the machine type selected, i.e.,

$$c_{ij} = b_{ij} + P_{ijh} - 1, \quad i = 0, 1, \ldots, I - 1; j = 0, 1, \ldots, J_i - 1 \text{ and } h \in H_{ij}. \tag{8}$$

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 Modeling of Batch Machine Types

As mentioned in Section 1, parts to be performed in a batch should be synchronized to have the same operation beginning and completion times. By viewing batches as virtual facilities, the difficult synchronization among these parts can be accomplished by synchronizing individual parts to batches. The *batch constraints* thus state that the total volume of parts beginning at time k for a batch operation may not exceed the total volume of batches beginning at time k, i.e.,

$$\sum_{i=0}^{I-1} \sum_{j \in B_i} v_{ij} \hat{\delta}_{ijhk} \le V_h \sum_{n=0}^{N_{hg}-1} \hat{\varphi}_{hgnk}, \; h \in H_B, \; g = 0, 1, \ldots, G_h - 1, \; k = 0, 1, \ldots, K-1, \tag{3}$$

where $\hat{\delta}_{ijhk}$ is a 0-1 *operation beginning variable*. It equals 1 if operation (i, j) *starts* on machine type h at time k (i.e., $b_{ij} = k$), and 0 otherwise. Similarly, $\hat{\varphi}_{hgnk}$ is a 0-1 *batch beginning variable*. It equals 1 if batch (h, g, n) *starts* at time k (i.e., $b_{hgn} = k$), and 0 otherwise.

Since batches act as a virtual parts that compete batch machines, the *machine capacity constraints for batch machines* state that the number of batches performed on a batch machine type h may not exceed the number of batch machines available, i.e.,

$$\sum_{g=0}^{G_h-1} \sum_{n=0}^{N_{hg}-1} \varphi_{hgnk} \le M_{hk}, \quad h \in H_B, k = 0, 1, \ldots, K-1, \tag{4}$$

where φ_{hgnk} is a 0-1 *batch in-processing variable*. It is 1 if batch (h, g, n) is performed on a machine of type h at time k, and 0 otherwise.

Similar to the operation in-processing variables δ_{ijhk} defined by (2), the operation beginning variables $\hat{\delta}_{ijhk}$, the batch beginning variables $\hat{\varphi}_{hgnk}$ and the batch in-processing variables φ_{hgnk} are not independent decision variables. They can be determined by the machine types for individual operations and operation beginning times.

The *batch precedence constraints* provide the processing sequence of batches on each machine, i.e.,

$$c_{hgn} + 1 \le b_{hg,n+1}, \quad h \in H_B, \; g = 0, 1, \ldots, G_h - 1, \tag{5}$$

where c_{hgn} is the completion time of batch (h, g, n), and $b_{hg,n+1}$ the beginning time of batch (h,g,n+1).

The *batch processing time requirements* state that each batch must be assigned the required amount of processing time P_{hg},

$$c_{hgn} = b_{hgn} + P_{hg} - 1, \quad h \in H_B, \; g = 0, 1, \ldots, G_h - 1, \; n = 0, 1, \ldots, N_{hg} - 1. \tag{6}$$

2.5 Objective

Various objective functions such as makespan have been used in literature. Study of practical scheduling, however, shows that the tardiness objective is likely to be more useful than other criteria such makespan (Blackstone et al., 1982). In addition, the additivity of the tardiness objective function facilitates the decomposition approach. The following objective function of weighted part tardiness and earliness is used to model the goal of on-time delivery and low work-in-process inventory.

$$\mathbf{J} \equiv \sum_{i=0}^{I-1} \left[W_i T_i^2 + \beta_i E_i^2 \right], \tag{9}$$

where $T_i = max\ [0, c_{i,J_i-1} - D_i]$ is the tardiness, and $E_i = max\ [0, \bar{b}_i - b_{i0}]$ the earliness. The coefficient W_i is the tardiness weight of part i, and β_i the earliness weight.

The overall problem is therefore to minimize the part tardiness and earliness penalty function (9), subject to the above machine capacity, batch, operation and batch precedence constraints. The decision variables are the machine types $\{h_{ij}\}$ for individual operations, operation beginning times $\{b_{ij}\}$, and batch beginning times $\{b_{hgn}\}$. Once these decision variables are determined, other variables $\{c_{ij}\}$, $\{\delta_{ijhk}\}$, $\{\hat{\delta}_{ijhk}\}$, $\{\varphi_{ijhk}\}$, and $\{\hat{\varphi}_{hgnk}\}$ can be easily derived. Among the constraints, machine capacity and batch constraints are "coupling constraints" as they couple together decision variables associated with different parts and batches. Since all the coupling constraints and the objective function are additive in terms of decision variables, the problem formulation is separable. Lagrangian relaxation can therefore be effectively used to solve it, as will be presented in the next Section.

3. SOLUTION METHODOLOGY

Lagrangian relaxation (LR) is a mathematical programming technique for performing constrained optimization. It has been successfully applied to various manufacturing systems to obtained good schedules with quantifiable quality in a reasonable amount of computation time (Hoitomt, et al., 1993; Liao et al., 1993; Luh et al., 1995). Recently, the embedding of dynamic programming within the Lagrangian relaxation framework for solving subproblems has improved algorithm convergence and schedule quality (Chen et al., 1995; Kaskavelis et al., 1995 and Wang et al., 1997).

Similar to the pricing concept of a market economy, the Lagrangian relaxation method replaces "hard" coupling constraints (e.g., the machine capacity constraints) by the payment of a certain "prices" (i.e., Lagrange multipliers) for the use of machines at individual time units. The original problem can thus be decomposed into many smaller subproblems. These subproblems are much

easier to solve as compared to the original problem, and solutions can be efficiently obtained by using "dynamic programming." These prices or multipliers are iteratively adjusted based on the degree of constraint violations following again the market economy concept (i.e., increase the prices for over-utilized time units and reduce the prices for under-utilized time units). 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 the dual costs (values of the dual function) are lower bounds to the optimal feasible cost. At the termination of such price updating iterations, a few constraints may still be violated when subproblem solutions are put together. Simple heuristics are thus applied to adjust subproblem solutions to form a feasible schedule satisfying all constraints. Heuristics can also be run after selected optimization iterations to check convergence status or to generate more candidate feasible schedules. Optimization and heuristics thus operate in a synergistic fashion to generate effective schedules. The quality of the feasible schedule can be quantitatively evaluated by comparing its cost to the largest lower bound provided by the dual function.

3.1 The Lagrangian Relaxation Framework

Standard machine capacity, batch machine capacity, and batch constraints are first “relaxed” by using nonnegative Lagrange multipliers $\{\pi_{hk},\ h \in H_{\overline{B}}\}$, $\{\pi_{hk},\ h \in H_B\}$ and $\{\gamma_{hgk}\}$, respectively. The Lagrangian is formed as:

$$L \equiv \sum_i \left[W_i T_i^2 + \beta_i E_i^2 \right] + \sum_{h \in H_{\overline{B}}} \sum_k \pi_{hk} \left(\sum_{i,j} \delta_{ijhk} - M_{hk} \right) + \sum_{h \in H_B} \sum_k \pi_{hk} \left(\sum_{g,n} \varphi_{hgnk} - M_{hk} \right)$$
$$+ \sum_{h \in H_B} \sum_g \sum_k \gamma_{hgk} \left(\sum_{i,j \in B_i} v_{ij} \hat{\delta}_{ijhk} - V_h \sum_n \hat{\varphi}_{hgnk} \right). \qquad (10)$$

In the above, $\{\pi_{hk},\ h \in H_{\overline{B}}\}$ are the prices for performing operations on standard machines, and $\{\pi_{hk},\ h \in H_B\}$ the prices for processing batches. The multipliers $\{\gamma_{hgk}\}$ are the prices for an operation to occupy the batch volume. With the multipliers given, the "relaxed problem" is to choose the decision variables to minimize the Lagrangian L subject to operation and batch precedence constraints. After regrouping relevant terms within L, the problem is decomposed into part and batch subproblems.

3.2 Part Subproblems and Their Resolutions

Part Subproblems

Collecting all the terms in (10) related to individual parts leads to the following part subproblems:

$$\min_{\{b_{ij}, h_{ij}\}} L_i \text{ with } L_i \equiv W_i T_i^2 + \beta_i E_i^2 + \sum_{j=0}^{J_i - 1} L_{ij}(b_{ij}, h_{ij}), \quad \forall i, \text{ with}$$

$$L_{ij}(b_{ij}, h_{ij}) \equiv \sum_{k=b_{ij}}^{c_{ij}} \pi_{h_{ij}k}, \quad \text{if } h_{ij} \in H_{\overline{B}}, \text{ and}$$

$$L_{ij}(b_{ij}, h_{ij}) \equiv v_{ij}\gamma_{h_{ij}gb_{ij}}, \quad \text{if } h_{ij} \in H_B, \tag{11}$$

subject to operation precedence constraints. The decision variables are the machine type $\{h_{ij}\}$ and beginning times $\{b_{ij}\}$ for individual operations of part i. Since multiplier $\pi_{h_{ij}k}$, $h_{ij} \in H_{\overline{B}}$ is the price for a operation to use a standard machine, and $\gamma_{h_{ij}gk}$ the price for an operation to occupy the batch volume, $L_{ij}(b_{ij}, h_{ij})$ is the utilization cost of operation (i, j). A part subproblem thus reflects a balance between part tardiness and earliness penalty and the utilization cost of operations.

Dynamic Programming for Part Subproblems

Dynamic programming (DP) is an effective method for solving multistage optimization problems, and can be implemented in the forward form as in Chen et al. (1995) or in the backward form. In either forward or backward form, DP stages correspond to individual operations, the states at a stage correspond to possible operation beginning times for each possible machine type of the operation. In this study, the backward DP is used to solve part subproblems for possible future extensions to deal with uncertainties (Luh, et al., 1997). It starts with the last operation (stage), and computes the following terminal cost for all possible machine type h_{i,J_i-1} and possible operation beginning time b_{i,J_i-1}:

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

The cumulative cost when moving backwards to the preceding stage is then obtained recursively according to the following DP equation subject to operation precedence constraints:

$$V_{ij}(b_{ij}, h_{ij}) = \min_{\{b_{i,j+1}, h_{i,j+1}\}} \{L_{ij}(b_{ij}, h_{ij}) + V_{i,j+1}(b_{i,j+1}, h_{i,j+1})\}$$

$$= L_{ij}(b_{ij}, h_{ij}) + \min_{\{b_{i,j+1}, h_{i,j+1}\}} V_{i,j+1}(b_{i,j+1}, h_{i,j+1}),\ 0 \le j < J_i - 1. \qquad (13)$$

The minimization in the above DP equation is to find the minimum cost for all possible machine types $\{h_{i,j+1}\}$ and beginning times $\{b_{i,j+1}\}$. The optimal subproblem cost L_i^* is obtained as the minimal cumulative cost at the first stage, i.e., $L_i^* = \min_{b_{i0}, h_{i0}} \beta_i E_i^2 + V_{i0}(b_{i0}, h_{i0})$. The optimal machine types and beginning times for individual operations can then be obtained by tracing forwards the stages. The computation complexity for the above backward DP algorithm is $O(K \Sigma_j |H_{ij}|)$.

3.3 Batch Subproblems and Their Resolutions

Batch Subproblems

By collecting all the terms in (10) related to batches on individual machines, batch subproblems are obtained as

$$\min_{\{b_{hgn}\}} L_{hg}, \text{ with } L_{hg} \equiv \sum_n \Big(\sum_{k=b_{hgn}}^{c_{hgn}} \pi_{hk} - \gamma_{hgb_{hgn}} V_h \Big), \quad \forall\, h \in H_B; g, \qquad (14)$$

subject to batch precedence constraints (5). The decision variables are batch beginning times $\{b_{hgn}\}$. Since multiplier π_{hk}, $h \in H_B$, is the price for the use of batch machines, and γ_{hgk} the price for the use of virtual facilities, a batch subproblem thus reflects a balance between batch machine utilization cost and the value for hosting operations.

DP for Batch Subproblems

Similar to part subproblems, a batch subproblem is a multi-stage optimization problem with each stage corresponding to a batch, and the states at a stage correspond to possible batch beginning times. The backward DP algorithm presented in Subsection 3.3 can thus be used to solve the batch subproblems.

3.4 Dual Problem and Lagrangian Multiplier Updating

Let L_i^* denote the minimal part subproblem cost of part i, and L_{hg}^* the minimal group subproblem cost for group g on machine type h, the high level Lagrangian dual problem is obtained as

$$\max_{\{\pi_{hk},\,\gamma_{hgk}\}} D, \quad \text{with} \quad D \equiv \sum_i L_i^* + \sum_{h\in H_B} \sum_g L_{hg}^* - \sum_{h,k} \pi_{hk} M_{hk}. \tag{15}$$

The Lagrangian dual function D is concave and piece-wise linear, and consists of many "facets" (Tomastik and Luh, 1993), with each facet corresponding to a set of decision variable values in the relaxed problem. Subgradient methods are commonly used to update the Lagrange multipliers in solving the dual problem because of their simplicity and the global convergence property. The methods update multipliers along the direction of subgradient[1] for some distance. The updating of multipliers requires the dual function to be evaluated many times. Each function evaluation involves solving all the subproblems once, and is extremely "expensive" for large problems. For example, it takes about 57% of total computation time for the Case 3 in Section 4 with 82 parts and 14 machines. To efficiently update multipliers, the interleaved subgradient (ISG) method was thus developed (Kaskavelis and Caramanis, 1995). Instead of solving all subproblems before updating multipliers, the ISG method updates multipliers after solving each subproblem. Numerical results show that the ISG method converges faster than a subgradient method, especially for problems of very large size. The algorithm convergence was established in Zhao, Luh and Wang (1997).

[1] The component of subgradient corresponding to a machine capacity constraint of standard machine type h at time k is $\sum_{i=0}^{I-1} \sum_{j\in B_i} \delta_{ijhk} - M_{hk}$.

Because of the combinatorial nature of the problem, the number of facets in the dual function increases drastically as the problem size increases, and the dual function approaches a smooth function. (Wang et al., 1997). This "smoothness" of the dual function motivates the use of optimization methods for smooth functions. Among these methods, conjugate gradient methods have attractive convergence properties and computation efficiency (Bertsekas, 1995). The conjugate directions are generated by

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

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 combining the "interleaved" concept with the conjugate gradient method, an interleaved conjugate gradient (ICG) method was 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.

Step 0: [Initialize.] Set iteration index n = 0. Given a set of initial multipliers, solve all the subproblems, and compute the dual cost and subgradient. Update multipliers along the direction of the subgradient with the step size computed as in subgradient methods (e.g., Bertsekas, 1995) by

$$\alpha^n = \gamma \frac{J^n - D^n}{(g^n)^T g^n}, \quad 0 < \gamma \le 2, \tag{17}$$

where J^n, α^n, D^n and g^n are respectively the lowest feasible cost, the step size, the dual cost and the subgradient at iteration n. Set subproblem index s = 1.

Step 1: [Solve a subproblem.] Solve subproblem s while keeping other subproblem solutions unchanged. Compute the *surrogate dual* cost according to (14) and *surrogate subgradient* with the current multipliers and the *latest available* solutions for individual subproblems.

Step 2: [Update multipliers.] Compute conjugate direction according to (16) with the surrogate subgradient, and update multipliers along the direction. The direction is reset as the surrogate subgradient after a given number of iterations. Since only one subproblem is solved for a set of multipliers, line search cannot be used to determine the step size. The step size is therefore computed according to (17).

Step 3: [Check stopping Criteria.] The maximum amount of computation time or the maximum number of iterations can be used as the stopping criteria. If the stopping criteria is satisfied, stop. Otherwise, increase s by one (Reset s to 1 if it is larger than the total number of subproblems), and go to S1.

3.4 Obtaining a Feasible Schedule

Since machine capacity and batch constraints have been relaxed, subproblem solutions, when put together, generally do not constitute a feasible schedule. A heuristic procedure is used to adjust subproblem solutions to form a feasible schedule. A list is created for batch operations within each group in the ascending order of their beginning times in subproblem solutions. Operations with beginning times ranging in a certain period are put into the same batch, subject to the volume of the batch. After a batch are formed, batch beginning time is decided according to operation beginning times and the operation precedence constraints.

A list of standard operations and batches is then created in the ascending order of their beginning times. The list heuristic presented in Hoitomt et al. (1993) is then extended to obtain a feasible schedule. Standard operations and the batches are scheduled on the required machine types according to the list as machines become available. If the capacity constraint for a particular machine type is violated at time k, a greedy heuristic based on the incremental change in **J** (the objective function in (9)) determines which operations or batches should begin at that time and which ones are to be delayed by one time unit. In computing the incremental change in **J**, higher priority is given to operations that are immediately followed by batch operations, since the delay of a batch causes delay

of all operations in the batch. The process repeats till the end of the list, and a feasible schedule is then obtained.

The cost **J** of the feasible schedule 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 $(\mathbf{J} - D)/D \times 100\%$ is often used as a quality measure of the feasible schedule.

4. NUMERICAL RESULTS

The method has been implemented using the object-oriented programming language C++. The testing of three cases is presented below to demonstrate the performance of the method developed. The first case considers the scheduling of a single batch machine, and the second a job shop with one batch machine and four standard machines. The testing of the two cases is to show that the algorithm provides high quality schedules by the integrated consideration of batch formation and sequencing. The third case is created based on the data from a practical job shop. The testing show that the algorithm is computationally efficient to solve practical problems. The resulting schedules of the three cases were obtained on a Pentium Pro200 personal computer.

Case 1.

In this case, 48 single operation parts with various due dates are to be scheduled on a single batch machine. According to their processing requirements, parts are classified into four groups. There are twelve parts in each of the first two groups, 15 parts in the third group, and nine parts in the fourth group. The processing times of the four groups of parts are 3, 4, 5 and 6, respectively. All parts have the same size of one, and the machine can process up to three parts simultaneously. All parts have the same tardiness weight of one, and there is no earliness penalty. Data of parts is summarized in Table 1.1. The scheduling time horizon is 100, and the total number of multiplier is 500.

Table 1.1. Input Data for Case 1

Part i	Group ID	Proc. Time	Due date		Part i	Group ID	Proc. Time	Due date
1	2	4	5		25	2	4	6
2	2	4	5		26	2	4	6
3	2	4	15		27	2	4	15
4	3	5	10		28	3	5	12

5	1	3	8		29	1	3	8
6	3	5	7		30	3	5	7
7	3	5	8		31	3	5	8
8	4	6	9		32	3	5	4
9	1	3	6		33	1	3	6
10	1	3	6		34	1	3	6
11	4	6	9		35	4	6	8
12	4	6	8		36	4	6	8
13	2	4	15		37	2	4	15
14	2	4	1		38	2	4	2
15	2	4	1		39	2	4	2
16	3	5	7		40	3	5	1
17	1	3	8		41	1	3	8
18	3	5	10		42	3	5	9
19	3	5	1		43	3	5	2
20	3	5	4		44	4	6	18
21	1	3	12		45	1	3	14
22	1	3	12		46	1	3	14
23	4	6	10		47	3	5	5
24	4	6	20		48	4	6	20

The feasible schedule shown in Table 2 is obtained in 47 CPU seconds. It has a cost of 44,142 with a relative duality gap 3.0%. According to the schedule, 16 batches of different groups are formed to process the 48 parts. Each batch is fully occupied with three parts of the same group. Since parts in each group are of the same size, optimal batch formation can be obtained by putting the parts into batches in the ascending order of part due dates (Uzsoy, 1995). The feasible schedule in Table 2 is therefore optimal.

Table 1.2. Feasible Schedule (A1) for Case 1

Parts in Batch	Group ID	Beginning - Completion
P14, P15, P38	2	0 - 3
P19, P40, P43	3	4 - 8
P39, P1, P2	2	9 - 12
P9, P10, P33	1	13 - 15
P34, P29, P41	1	16 - 18
P5, P17, P21	1	19 - 21
P25, P26, P27	2	22 - 25
P22, P45, P46	1	26 - 28
P20, P32, P47	3	29 - 33
P6, P16, P30	3	34 - 38
P7, P31, P42	3	39 - 43

P37, P3, P13	2	44 - 47
P4, P18, P28	3	48 - 52
P12, P35, P36	4	53 - 58
P8, P11, P23	4	59 - 64
P24, P44, P48	4	65 - 70

Case 2.

In this case, there are two standard machine types (M1 and M2) each with two machines, and one batch machine (M3). All machines are available across the time horizon. Ten parts with various due dates are to be scheduled, and are available from time 0. The tardiness weights for all parts are one, and there is no earliness penalty. Each part may consist of one or two standard operations and a batch operation. The batch operations belong to two groups. The sizes of operations in the two groups are three and two, respectively, and the batch volume is six. Data of the ten parts is shown in Table 2.1. The scheduling time horizon is 50, and the total number of multipliers is 250.

The feasible schedule (S1) has a cost of 561 with a duality gap 1%, and is obtained within 17 seconds. When keep on running the algorithm to 60 seconds, the feasible cost remains unchanged, and a dual cost of 560.98 is obtained. Since dual cost serves as a lower bound to the optimal feasible cost, and feasible cost should be an integer value here, the schedule obtained is optimal. The resulting operation beginning and completion times are shown in Table 2.2, where M11 and M12 are the two identical machines in machine type M1, and M21 and M22 the two machines in M2. The Gannt chart of the feasible schedule is shown in Figure 1, where four batches are formed and fully occupied by the ten batch operations, and the batch machine is the bottleneck. Parts are processed in batches on batch machine, and are processed individually on standard machines, as shown by the shadowed boxes in the Gannt chart.

To illustrate the advantage of explicit modeling batch machines, the problem is re-solved by treating the batch machine as a standard machine in the optimization model, but considering batch formation in heuristics. The resulting schedule (S2) for this simplified model has a cost 585, which is 4% higher than that of schedule S1.

Table 2.1. Data for Case 2

Part i	D_i	Op.(i, j)	h	P_{ijh}	Group

0	2	(0, 0)	M1	2	
		(0, 1)	M2	2	
		(0, 2)	M3	4	1
1	2	(1, 0)	M2	3	
		(1, 1)	M1	1	
		(1, 2)	M3	4	1
2	4	(2, 0)	M2	1	
		(2, 1)	M3	4	1
		(2, 2)	M1	1	
3	4	(3, 0)	M1	1	
		(3, 1)	M3	4	1
		(3, 2)	M2	3	
4	6	(4, 0)	M1	2	
		(4, 1)	M3	5	2
5	6	(5, 0)	M1	1	
		(5, 1)	M3	5	2
		(5, 2)	M2	2	
6	6	(6, 0)	M2	3	
		(6, 1)	M3	5	2
7	8	(7, 0)	M1	4	
		(7, 1)	M3	5	2
8	8	(8, 0)	M2	2	
		(8, 1)	M3	5	2
9	8	(9, 0)	M1	4	
		(9, 1)	M3	5	2

Table 2.2. Schedule (S1) for Case 2

Part i	Op.(i, j)	Machine	b_{ij} - c_{ij}
0	(0, 0)	M11	0 - 1
	(0, 1)	M21	3 - 4
	(0, 2)	M3	5 - 8
1	(1, 0)	M21	0 - 2
	(1, 1)	M11	3 - 3
	(1, 2)	M3	5 - 8
2	(2, 0)	M22	0 - 0
	(2, 1)	M3	1 - 4
	(2, 2)	M11	5 - 5
3	(3, 0)	M12	0 - 0
	(3, 1)	M3	1 - 4
	(3, 2)	M21	5 - 7
4	(4, 0)	M12	1 - 2
	(4, 1)	M3	9 - 13
5	(5, 0)	M11	2 - 2
	(5, 1)	M3	9 - 13
	(5, 2)	M21	14 - 15
6	(6, 0)	M22	3 - 5
	(6, 1)	M3	9 - 13
7	(7, 0)	M11	6 - 9
	(7, 1)	M3	14 - 18
8	(8, 0)	M22	1 - 2
	(8, 1)	M3	14 - 18
9	(9, 0)	M12	3 - 6
	(9, 1)	M3	14 - 18

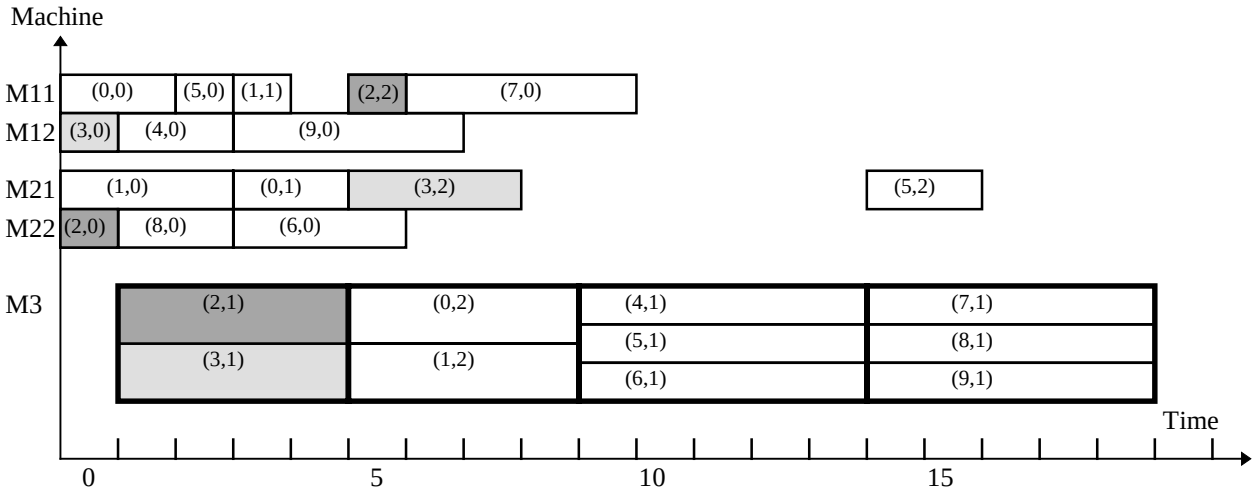

Figure 1: Gantt Chart of Feasible Schedule for Case 2

To show the effect of batch processing time on the feasible schedule, Case 2 is modified by doubling all batch operation processing times. By running the algorithm for 34 seconds, a feasible schedule (S3) with a cost of 4,062 and a duality gap of 6.11% is obtained. In the schedule, batches are formed exactly the same as in schedule S1, but have different processing sequence and beginning times. Since the batch machine is the bottleneck, this data change leads to significant increase of the feasible cost.

Case 3.

This case is created based on the data from a practical job shop. There are four standard machine types with a total of seven machines, and four batch machine types with seven machines. Some machines may not be available during certain periods on the time horizon. Eighty two parts with a total of 752 operations are to be scheduled. Parts have various due dates with two level of tardiness weights (1.0 and 0.5), and there is no earliness penalty. Operations on batch machine types belong to 16 groups with sizes of two or three, and the volume of a batch is six. Operation processing times range from 1 to 24, and part due dates are scattered from 0 to 230. The scheduling time horizon is 2046.

The resulting feasible schedule has a cost of 39,944 with a duality gap 33%, and is obtained within 600 CPU seconds. Results at some selected iterations are summarized in Table 3 to show the improvement on both feasible schedule and lower bound as the number of iterations increases. The results show that the method developed keeps on improving schedule quality by reducing the feasible cost and improving the lower bound by maximizing the dual function. A good schedule is obtained within 100 iterations or about 2.5 minutes.

Table 3. Testing Results for Case 3

Iteration	Feasible Cost	Dual Cost	Duality Gap	CPU seconds
0	61108	156	38947%	1
100	42033	26805	57%	157
200	40240	28720	40%	310
300	39944	29627	35%	453
400	39944	29941	33%	597

A few practically used performance metrics are provided 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 the cell.

Average work-in-process inventory: the average number of parts in the shop over the time horizon.

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

Total processing time/Total lead time: the sum of all operation processing times / the sum of all part lead times

The performance metrics of the feasible schedules at some selected iterations are summarized in Table 4. They also show that the method developed keeps on improving the schedule quality at the number of iterations increases.

Table 4. Performance Metrics of Feasible Schedule

Iteration	0	100	200	400
Makespan	1065	1052	1062	1047
Maximum Work-in-process Inventory	47	35	34	34
Average work-in-process inventory	27.04	19	17.78	18.02
Average lead time	392	273	258	257
Total processing time/Total lead time (%)	25	35	37	37

By reducing the number of batch machines to one machine per type, the schedule obtained has a cost of 41,511 with a duality gap of 39% at 600 seconds. It shows that after reducing the number of batch machines, the feasible cost significantly increased. There are more competitions on the batch machines, and the method will take more computation time to find a good schedule.

5. CONCLUSIONS

In this paper, the scheduling of job shops with batch machines is considered in an integrated fashion to decide both batch formation and sequencing. A novel "separable" integer programming formulation is developed with manageable numbers of variables and constraints. A Lagrangian relaxation based method is then applied to generate high quality solutions with quantifiable quality.

Through the iterative problem resolution, the method keeps on improving the schedule quality as indicated by the feasible costs and performance metrics. Numerical testing shows that the integrated consideration of batch formation and sequencing results in high quality schedules, and the algorithm is computationally efficient to solve practical problems.

Appendix A. A List of Symbols

δ_{ijhk}: 0-1 operation-level variable which is one if operation (i, j) is performed on machine type h at time k, and zero otherwise.

$\hat{\delta}_{ijhk}$: 0-1 operation level variable which is one if operation (i, j) starts on machine type h at time k, and zeor otherwise.

φ_{hgnk}: 0-1 batch-level variable which is one if batch (h, g, n) is being processed at time k, and zero otherwise.

$\hat{\varphi}_{hgnk}$:0-1 batch level variable which is one if batch (h, g, n) starts at time k, and zero otherwise.

b_{ij}: beginning time of operation (i, j)

b_{hgn}: beginning time of batch (h, g, n)

c_{ij}: completion time of operation (i, j)

c_{hgn}: completion time of batch (h, g, n)

D_i: due date of part i

g: group index variable

h: machine type variable, $h \in H$

H: set of all machine types

H_B: set of all batch machine types

$H_{\overline{B}}$: set of all standard machine types

H_{ij}: set of machine types capable of performing operation (i, j)

J: objective function to be minimized

J_i: number of operations for part i

k: discretized time index

K: time horizon

M_h: number of identical machines in machine type h

M_{hk}: capacity of machine type h at time k

N_{hg}: number of batches in group g of machine type h

P_{ijh}: processing time of operation (i, j) on machine type $h \in H$

P_{hg}: processing time of a batch within group g of machine type h

T_i: tardiness of part i, defined as $T_i = max[0, c_{i,J_i-1}-D_i]$

v_{ij}: size of part i when performing operation j

V_{hg}: capacity of a batch in group g on machine type h

W_i: part tardiness weight

Acknowledgments. This work was supported in part by the National Science Foundation under Grant DMI-9500037, and the Advanced Technology Center for Precision Manufacturing, the University of Connecticut. The authors would like to thank Dr. Debra J. Hoitomt of United Airlines for her earlier work on this problem.

REFERENCES

1. Ahmadi, J. H., Ahmadi, R. H., Dasu, S., Tang, C. S., 1992, "Batching and Scheduling Jobs on Batch and Discrete Processors," *Operations Research*, Vol. 40, pp. 750-763.
2. Blackstone, J.H., D.T. Phillips and G.L. Hogg, 1982, "A State-of-the-art Survey of Dispatching Rules for Manufacturing Job Shop Operations," *International Journal of Productions Research*, Vol. 20, pp. 27-45.
3. Chen, H., Chu, C., Proth, J. M., 1995, "A More Efficient Lagrangian Relaxation Approach to Job Shop Scheduling Problems," *Proceeding of IEEE Conference on Robotics and Automation*, pp. 496-501.
4. Dessouky, Y. M., Roberts C. A., Dessouky M. M. and Wilson G., 1996, "Scheduling Multipurpose Batch Plants with Junction Constraints," *International Journal of Productions Research*, Vol. 2, pp. 525-541.
5. Hoitomt, D. J., Luh, P. B., Pattipati, K. R., 1993, "A Practical Approach to Job-Shop Scheduling Problems," *IEEE Transactions on Robotics and Automation*, Vol. 9, pp. 1-13.
6. Hiriart-Urruty, J. B. and Lemarechal, C., 1993, *Convex Analysis and Minimization Algorithms, I and II* (Springer-Verlag, Berlin).
7. Kaskavelis, C.A. and Caramanis M.C., 1995, "Efficient Lagrangian Relaxation Algorithms for Real-life-size Job-shop Scheduling Problems," Working Paper, Boston University, Department of Manufacturing Engineering.
8. Lee, C.Y., S.D. Liman and A. Wirakusumah, 1993, "Product Batching and Batch Sequencing for NC Punch Presses," *International Journal of Productions Research*, Vol. 31, pp. 1143-1156.
9. Lemarechal, C., 1978, "Bundle Methods in Nonsmooth Optimization," *Proceedings of a IIASA Workshop, ed. C. Lemarechal and R. Mifflin* Pergamon Press, Great Britain, 1978), pp. 77-82.
10. Liao, D. Y., Chang, S. C., Yen, S. R., Chien, C. C., 1993, "Daily Scheduling for R&D Semiconductor Fabrication," *Proceeding of IEEE Conference on Robotics and Automation*, pp. 77-82.
11. Luh, P.B., D. Chen and L.S. Thakur, 1997, "Modeling Uncertainty in Job Shop Scheduling," *Proceedings of the First International Conference on Operations and Quantitative Management*, Jaipur, India, pp. 490-497.
12. Luh, P.B., L. Gou, T. Odahara, M. Tsuji, K. Yoneda, T. Hasegawa and Y. Kyoya, 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.
13. Tomastik, R. N. and Luh, P. B., 1993, "The Facet Ascending Algorithm for Integer Programming Problems," *Proceedings of 32nd IEEE Conference on Decision and Control* (San Antonio, Texas, USA), pp. 2880-2285.
14. Tomastik, R. N. and Luh, P. B., 1996, "A Reduced-Complexity Bundle Method for Maximizing Concave Nonsmooth Functions," *Proceedings of the 35th IEEE Conference on Decision and Control* (Kobe, Japan).
15. Uzsoy, R., 1995, "Scheduling Batch Processing Machines with Incompatible Job Families," *International Journal of Productions Research*, Vol. 33, pp. 2685-2708.

16. Wang, J., Luh, P.B., 1994, "Optimization Based Scheduling of a Batch Processing Facility," *Proceedings of the Conference on Computer Integrated Manufacturing in the Process Industries*, (New Brunswick, NJ, USA), pp. 3-20.
17. Wang, J.H., Luh, P.B., X. Zhao and J.L. Wang, 1997, "An Optimization-based Algorithm for Job Shop Scheduling," Special Issue of SADHANA on Competitive Manufacturing Systems, Indian Academy of Sciences, Bangalore, India.
18. Webster S., and Baker K. R., 1995, "Scheduling Groups of Jobs on a Single Machine," *Operations Research*, Vol. 43, pp. 692-703.
19. Zijm, W. H. M., 1995, "The Integration of Process Planning and Shop Floor Scheduling in Small Batch Part Manufacturing," Annals of the CIRP, Vol. 44, pp. 429-432.
20. Zhao, X., P.B. Luh and J. Wang, 1997, "The Surrogate Gradient Algorithm for Lagrangian Relaxation Method," Submitted to The 36th IEEE Conf. on Deci & Cont., San Diego, CA, USA.