Annals of Operations Research 76(1998)233–259

Job shop scheduling with group-dependent setups, finite buffers, and long time horizon

Peter B. Luh, Ling Gou and Yuanhui Zhang
Department of Electrical and Systems Engineering, University of Connecticut, Storrs, CT 06269-2157, USA

Takaaki Nagahora and Makoto Tsuji
Toshiba HamaKawasaki Works, 2-1, Ukishima-cho, Kawasaki-ku, Kawasaki-shi, 210 Japan

Kiyoshi Yoneda, Tetsuo Hasegawa, Yuji Kyoya and Toshiyuki Kano
Systems & Software Engineering Laboratory, Toshiba Corporation, 70, Yanagi-cho, Saiwai-ku, Kawasaki-shi, 210 Japan

Scheduling is a key factor for manufacturing productivity. Effective scheduling can improve on-time delivery of products, reduce inventory, cut lead times, and improve the utilization of bottleneck resources. This study was motivated by the design and implementation of a scheduling system for the manufacturing of Toshiba's gas insulated switchgears. The manufacturing is characterized by significant machine setup times, strict local buffer capacities, the option of choosing a few alternative processing routes, and long horizon as compared to the time resolution required. This problem has been recognized to be extremely difficult because of the combinatorial nature of integer optimization and the large size of the real problem. Our goal is thus to obtain near-optimal schedules with quantifiable quality in a computationally efficient manner. To achieve this goal, a novel integer optimization formulation with a separable structure is developed, and a solution methodology based on a combined Lagrangian relaxation, dynamic programming, and heuristics is developed. The method has been implemented using the object-oriented programming language C++, and numerical testing shows that the method generates high-quality schedules in a timely fashion to achieve on-time delivery of products and low inventory. Through explicit consideration of setups, tanks with the same processing requirements tend to be processed together to avoid excessive setups. The integrated treatment of machines and buffers facilitates the smooth flow of parts through the system. The embedded routing selection mechanism also balances the load among candidate routes. Finally, the newly developed "time step reduction technique" implicitly establishes two time scales to reduce computational requirements without much loss of modeling accuracy and scheduling performance, thereby enabling the resolution of long horizon problems with controllable computational requirements.

Keywords: job shop scheduling, optimization-based scheduling, Lagrangian relaxation, machines with setups, finite buffers, long time horizon

1. Introduction

Scheduling is a key factor for manufacturing productivity. Effective scheduling can improve on-time delivery of products, reduce inventory, cut lead times, and improve the utilization of bottleneck resources. Scheduling is especially critical for multi-product and small-lot production, where many products are required to be manufactured in small quantities to meet the diverse and likely time-varying demand. Different manufacturing requirements of various products and the existence of multiple and time-varying bottleneck resources make the scheduling task extremely difficult.

This study was motivated by the design and implementation of a scheduling system for the manufacturing of Toshiba's gas insulated switchgears (GIS). A GIS consists of inter-connecting aluminum or steel tanks filled with insulating gas SF_6, and is an advanced circuit-breaking device used by electric utilities in their control of high voltage/high current electricity flow. A simplified tank, the basic component of a GIS, is as shown in figure 1.

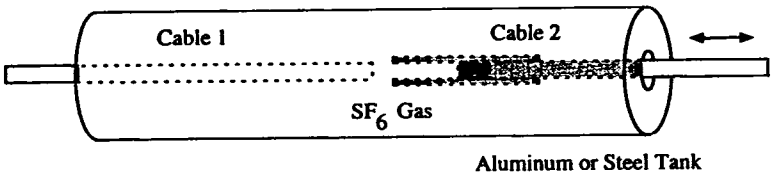

Figure 1. A simplified tank.

The manufacturing facility is arranged as a flow shop since most tanks go through similar production stages with the possibility of skipping certain ones. For some machine types, setup times are either negligible or are independent of the tank processing sequence, and can be lumped with processing times. These machine types are called "standard" machine types. For other machine types, the setup time is sequence dependent, however, with the following feature. The setup time between processing two "similar" tanks is much smaller than the setup time between two "different" tanks, where similarity is determined by the selected field(s) of tanks' Group Technology code. The fields of interest depend upon machine type, since while tank diameters may be important for one machine type, lengths may be important for another. This kind of setups will be referred to as "group-dependent setups". There are finite capacity local buffers between production stages, and a tank may have the option of going through a few alternative (exclusive-or) routes for processing. The scheduling horizon can range from a few weeks to a few months, and the time resolution required is 1 hour. The above description is typical for a large number of manufacturing facilities.

The scheduling problem has been recognized to be extremely difficult because of the combinatorial nature of integer optimization and large sizes of the real problem. It is often difficult to obtain optimal schedules, especially within a limited amount of computation time. The goal of this study is thus to develop a method that can generate

near-optimal solutions with quantifiable quality in a computationally efficient manner. To achieve this, a novel integer optimization formulation considering all the above mentioned features is developed in section 3. The formulation is "separable" in the sense that the objective function and all "coupling" constraints are additive in terms of basic decision variables. In section 4, a solution methodology based on the combined Lagrangian relaxation technique and heuristics is developed following [1]. Within the Lagrangian relaxation framework, dynamic programming is used for subproblem solving to have better convergence and to obtain higher dual costs as in [2,25]. The special features of the system considered here, however, complicate the formation and resolution of subproblems.

The method has been implemented using the object-oriented programming language C++, and numerical testing shows that the method generates high-quality schedules in a timely fashion to achieve on-time delivery of products and low inventory. Through explicit consideration of setups, tanks with the same processing requirements tend to be processed together to avoid excessive setups. The integrated treatment of machines and buffers facilitates the smooth flow of tanks through the system. The embedded routing selection mechanism also balances the load among candidate routes. Finally, the newly developed "time step reduction technique" implicitly establishes two time scales to reduce algorithm complexity, thereby enabling the resolution of long horizon problems with controllable computational requirements.

2. Literature review

Industrial evidence and academic research suggest that effective managing of sequence-dependent setups is one of the critical factors to improve manufacturing system performance [3]. In a survey of industrial schedulers, 70% of the schedulers reported that they had to deal with sequence-dependent setups, and almost all named meeting due dates as their most important criterion [4]. Most studies on sequence-dependent setups, however, focused on either a single machine or several identical machines (for example [5–7]). Only a few studies addressed sequence-dependent job shops or flow shops (for example [8,9]). Total setup time, total setup cost, or makespan are the most frequently used objectives, and branch-and-bound and heuristics are the most widely used methods. For example, minimizing total setup time for a single machine is proved to be equivalent to the well-known traveling salesman problem, which can be solved by a few heuristic algorithms [10]. Sequence-dependent two-stage flow shops with a makespan objective have been studied in [11], and sequence-dependent job shops with total setup time as the objective have been studied in [9], both by using a branch-and-bound method. The computation time of the branch-and-bound method, however, increases exponentially with the problem size [11].

Several authors considered due date information in their studies. A class of heuristic procedures to minimize maximum lateness for identical machines with sequence-dependent setups and dynamic job arrivals was presented in [12]. Various due date

related heuristics were investigated for sequence dependent job shops in [13,14], and the impact of setups was discussed. The minimum tardiness problem with sequence-dependent setups is believed to be NP-hard, and a genetic algorithm for a single machine was presented in [15]. The understanding of sequence-dependent setups, however, is still believed to be far from complete, especially for scheduling to meet due dates.

Although finite buffers are a reality in organizations and could be one of the limiting factors in system performance, most scheduling problems do not consider them. Job shops with finite buffers were addressed in [16] based on timed Petri-net simulation and constrained propagation heuristics.

Scheduling to meet due dates in the presence of group-dependent setups and finite buffers is addressed in this paper. Instead of using heuristics, an optimization-based method is developed to obtain near-optimal schedules with quantifiable quality in a computationally efficient manner.

3. Problem formulation

Although machines are essentially arranged as a flow shop, tanks may skip one or more stages based on processing requirements. The sequencing of tanks at various stages can also be altered. From the scheduling viewpoint, the problem is more like a job shop than a flow shop. The formulation to be presented next is built on our previous work on job shop scheduling [1] with the following new features: introducing the concept of a "run" to describe the setup status of a machine; treating finite buffers as machines with given capacity but unspecified processing times (creating artificial "buffering" operations); selecting a processing route among alternative ones; and using "time step reduction" to implicitly establish two time scales to reduce computational requirements without much loss of modeling accuracy and scheduling performance. The terms "tank" and "part" will be used interchangeably, and a list of symbols used in this paper is provided in the appendix for reference.

3.1. General description and terminology

Time step reduction. In view of the long scheduling horizon as compared to the time resolution required, the "time step reduction technique" is developed. Assume that the horizon is divided into T "resolution increments" indexed by t, $0 \le t \le T-1$, and R resolution increments aggregate into an "enumeration step" indexed by k, $0 \le k \le K$. For example, a 500-hour horizon can be divided into 500 one-hour resolution increments, and 10 one-hour resolution increments aggregate into a ten-hour enumeration step (a shift for the GIS facility). An operation requiring 6 hours on a machine is approximately represented as occupying 60% of a ten-hour enumeration step (or a fraction of two consecutive enumeration steps) – fractional but quantized machine utilization as shown in figure 2. Multiple "short" operations are thus allowed to "share"

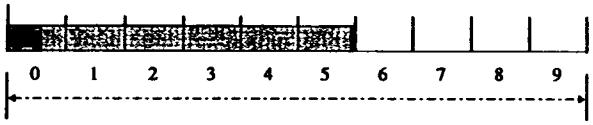

Figure 2. Enumeration step versus resolution increments, where 10 one-hour resolution increments aggregate into a ten-hour enumeration step, and a 6-hour operation occupies 60% of an enumeration step.

a machine within an enumeration step, and a part with several "short" operations is also allowed to flow through the machines within a single enumeration step. All input data are specified in terms of resolution increments. As will be presented later, the complexity of the method depends significantly on the number of enumeration steps. With the proper selection of the number of resolution increments to be contained within an enumeration step, computational requirements can be controlled (see [17] for an earlier version of the technique).

Machines and buffers. There is a set of machine types H, and each machine type may consist of one or multiple identical machines. The number of machines available per type at each resolution increment is a given integer. The average number of type h machines available at enumeration step k, denoted as M_{kh}, is thus a quantized fraction. For simplicity of analysis, each machine with group-dependent setups is modeled as a machine type of its own, and the set of machine types with group-dependent setups is denoted by H_s.

There are two categories of finite capacity local buffers. For some buffers, the number of tanks that can be held is a constant, e.g., one tank regardless of tank size. A machine with two pallets is an example. For other buffers, the number depends on tank size, e.g., three small tanks or two large tanks. Regardless of its category, a finite capacity buffer is modeled as a machine type, with M_{kh} representing the buffer capacity. The parameter M_{kh} can be a quantized fraction for a buffer that holds a fixed number of tanks, or can be a real number for a buffer characterized by its volume.

Tanks. There are I tanks to be scheduled. Tank i, $0 \le i \le I-1$, has a given earliest beginning time a_i (arrival time of raw materials), a desired release time $\bar{b}_i$, and a due date d_i. It has to go through a sequence of operations according to a specified process plan, with several simple fork-join type of alternative (exclusive-or) routes along the way. A simplified process plan is depicted in figure 3, where the jth operation of tank i is identified as operation (i, j), $0 \le j \le J_i - 1$. For simplicity of presentation, the first and the last operations which tank i must go through are denoted as $(i, 0)$ and $(i, J_i - 1)$, respectively. Operation (i, j) has to be performed by a machine belonging to an eligible type $h \in H_{ij}$ with given processing time t_{ijh}. It is assumed that standard machine types and machine types with setups do not co-exist in a single H_{ij}.

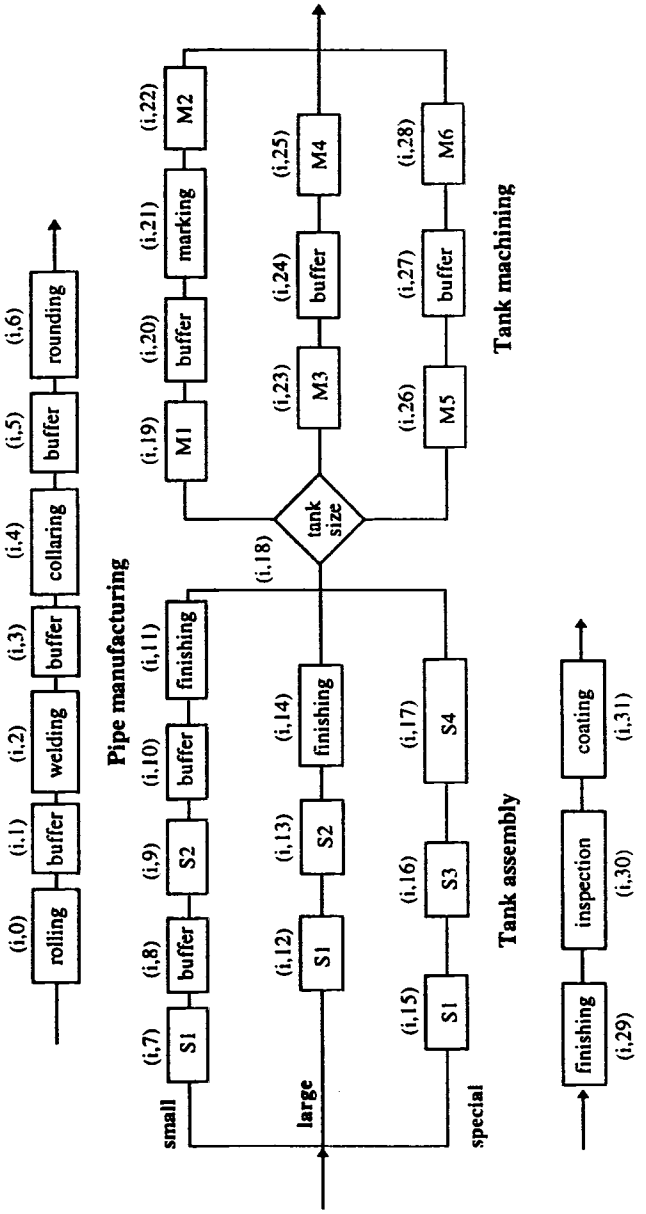

Figure 3. A simplified tank process plan, where simple fork-join type of alternative routes are depicted. For example, after rounding operation $(i,6)$, a small tank can select either one of the three routes, i.e., small, large, and special lines, for its further processing.

A tank in a buffer is considered as undergoing a "buffering" operation, e.g., operations $(i,1)$ and $(i,3)$ in figure 3. A buffering operation begins at the completion of its preceding manufacturing operation, and ends at the start of its subsequent manufacturing operation, with processing time a decision variable to be optimized. A "dummy" operation is introduced at a juncture which is both a fork node and a join node (e.g., $(i,18)$). The set of possible immediate subsequent operations is denoted by I_{ij}, and is usually a singleton except for a fork node.

3.2. *Modeling of standard machine types and buffers*

The *machine capacity constraints* for standard machine types and buffers state that the "average capacity" of a machine type cannot be exceeded at each enumeration step, i.e.,

$$\sum_{i,j} \left[v_{ij} \delta_{ijkh} \right] + m_{kh} = M_{kh}, \qquad \forall k, h \in H_s. \tag{1}$$

In the above, δ_{ijkh} represents the "fraction" of the enumeration step k that operation (i,j) is active on machine type h, and is a quantized fraction with a value ranging between 0 and 1; and m_{kh} is a non-negative slack variable satisfying $0 \le m_{kh} \le M_{kh}$. The value v_{ij} equals 1 for a buffer that can hold a fixed number of tanks, and v_{ij} represents tank volume for buffers that can hold varying numbers of tanks depending on tank sizes. For a manufacturing operation, $v_{ij} \equiv 1$.

3.3. *Modeling of machines with group-dependent setups*

It is well known that scheduling problems with sequence-dependent setups are extremely difficult to deal with, and a separable problem formulation without introducing an excessive number of variables is unlikely to be obtained. Although the setup requirements considered here are sequence dependent, the special "group dependency" is exploited by following [18] to obtain a separable problem formulation.

All tanks that can be assigned to a machine h with group-dependent setups are classified into total G_h "groups" based on the appropriate field(s) of tanks' Group Technology code. The setup time required in-between processing two tanks of the same group is small, and can generally be ignored. A significant setup, however, is required in-between processing two tanks of different groups, with exact time determined by the group of the second tank. Tanks of a particular group processed under a single setup form a "run". These concepts are shown in figure 4, where each group has two runs.

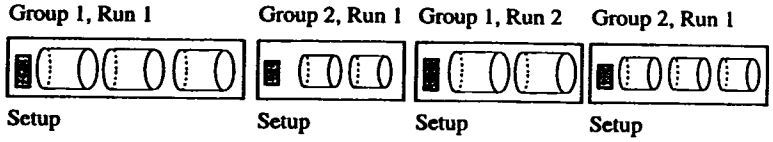

Figure 4. Group-dependent setup concept, where each group has two runs.

Suppose that all tanks belonging to group g, $1 \le g \le G_h$, are processed in N_{hg} runs, with the nth run denoted as run (h, g, n), where n is the run ID, and $1 \le n \le N_{hg}$. Without loss of generality, the number N_{hg} is assumed given. To start processing run (h, g, n), a setup with duration S^r_{hg} is required. The beginning, processing, and completion times of a run are denoted as b^r_{hgn}, t^r_{hgn}, and c^r_{hgn}, respectively. With any two of them treated as decision variables, the third is uniquely specified. In this study, b^r_{hgn} and c^r_{hgn} are selected as decision variables. The key to obtain a separable formulation is the translation of the complicated setup requirements explained above to an equivalent one: "a tank should be processed when one of the appropriate runs is active." This condition is then described by the following "group constraints" in linear inequality form.

The *group constraints* state that a tank cannot be processed unless one of the appropriate runs is active, i.e.,

$$\sum_{i,j} \delta_{ijkh} \le \sum_{n} \varphi_{khgn} - \sum_{n} \xi_{khgn}, \quad \forall k, h \in H_s, g.$$

In the above, φ_{khgn} is the fraction of an enumeration step that run (h, g, n) is active on machine type h at enumeration step k; ξ_{khgn} is the fraction of the enumeration step k that run(h, g, n) is being set up on machine type h, where setup is assumed to occur during the time interval $[b^r_{hgn}, (b^r_{hgn} + S^r_{hg} - 1)]$. The above inequality constraint can be rewritten as the following equality constraints:

$$\sum_{i,j} \delta_{ijkh} + \sum_{n} \xi_{khgn} + \sigma_{khg} = \sum_{n} \varphi_{khgn}, \quad \forall k, h \in H_s, g, \tag{2}$$

where σ_{khg} is a non-negative slack variable satisfying $0 \le \sigma_{khg} \le N_{hg}$.

Special considerations are needed to model the initial state of a machine in H_s. At time 0, the machine state could be in one of four possible cases: has been set up for a tank (say tank 1) of a particular group, just finished the setup for the tank, is processing the tank, or just finished processing the tank. To model these situations, a run for that group is assumed to have started at a negative time. The run should include at least one tank, and after processing tank 1, the run can continue to process other tanks within the group without setup, or to start a setup for another group. In either case, the run beginning time and the operation beginning time of tank 1 are treated as given (except for the last case where tank 1 is just finished), and the run completion time is a decision variable to be optimized subject to having at least one tank within the run.

Machine capacity constraints with respect to runs state that the average capacity of a machine type with respect to runs cannot be exceeded, i.e.,

$$\sum_{g,n} \varphi_{khgn} + \phi_{kh} = M_{kh}, \quad \forall k, h \in H_s, g. \tag{3}$$

In the above, ϕ_{kh} is a non-negative slack variable satisfying $0 \le \phi_{kh} \le M_{kh}$. Since each machine type with group-dependent setups consists of a single machine, M_{kh} is a quantized fraction in-between 0 and 1.

Run processing time requirements state that the completion time of a run equals the beginning time plus the required processing time, i.e.,

$$c^r_{hgn} = b^r_{hgn} + t^r_{hgn} - 1, \quad \forall(h, g, n). \tag{4}$$

Without loss of generality, the runs of a particular group are assumed to be processed in the ascending order of their run IDs, and this can be represented as the following *run sequencing constraints*:

$$c^r_{hgn} + 1 \le b^r_{hg(n+1)}, \quad \forall(h, g, n). \tag{5}$$

3.4. Operation precedence constraints and processing time requirements

Operation precedence constraints require the completion time of an operation plus a required "timeout" to be smaller than or equal to the beginning time of the immediate subsequent operation, i.e.,

$$c_{ij} + S_{ijl} + 1 \le b_{il}, \quad \forall(i, j), \text{ with } (i, l) \in I_{ij}. \tag{6}$$

In the above, b_{il} and c_{ij} represent the beginning time of (i, l) and the completion time of (i, j), respectively; (i, l) is the selected immediate subsequent operation of (i, j). The parameter S_{ijl} is the required "timeout" between (i, j) and (i, l), representing processes not explicitly modeled in the problem formulation (e.g., the transportation time required in-between the two operations). The above inequalities are replaced by equalities if either (i, j) or (i, l) is a buffering operation, since a buffering beginning time is the preceding operation's completion time plus the required timeout, and the completion time is the subsequent operation's beginning time. For clarity of presentation, a tank's beginning and completion times are defined as its first operation's beginning time b_{i0} and its last operation's completion time c_{i,J_i-1}, respectively.

Arrival time constraints state that the processing of a tank cannot be started before the arrival of the raw material, i.e.,

$$a_i \le b_{i0}, \quad \forall i. \tag{7}$$

Processing time requirements state that the completion time of an operation equals the beginning time plus the required processing time, i.e.,

$$c_{ij} = b_{ij} + t_{ijh} - 1. \quad \forall(i, j) \text{ and } h \in H_{ij}. \tag{8}$$

As mentioned, processing times for manufacturing operations are given, but the time required for a buffering operation is unspecified.

3.5. Objective function

The objective of scheduling is to ensure on-time product delivery while keeping low work-in-process inventory. This is represented by minimizing the sum of weighted quadratic penalties for missing tank due dates and penalties for releasing raw materials too early:

$$J \equiv \sum_i \left[w_i T_i^2 + \beta_i E_i^2 \right]. \tag{9}$$

In the above, T_i is the tardiness of part i defined as the amount the part completion time c_{i,J_i-1} passes its given due date d_i in the enumeration step, i.e.,

$$T_i \equiv \left(\left\lfloor \frac{c_{i,J_i-1}}{R} \right\rfloor - \left\lfloor \frac{d_i}{R} \right\rfloor \right)^+,$$

where $\lfloor x \rfloor$ is the "floor function" returning the integer portion of x, and $(x)^+ \equiv \max[0, x]$. The part earliness, E_i, is similarly defined as the amount the part beginning b_{i0} leads the desired part release time $\bar{b}_i$, i.e.,

$$E_i \equiv \left(\left\lfloor \frac{\bar{b}_i}{R} \right\rfloor - \left\lfloor \frac{b_{i0}}{R} \right\rfloor \right)^+.$$

Parameters w_i and β_i are weights associated with those penalty terms. The above penalties define a time window in which the part can be scheduled without penalty. The penalties for a two-operation part are shown in figure 5.

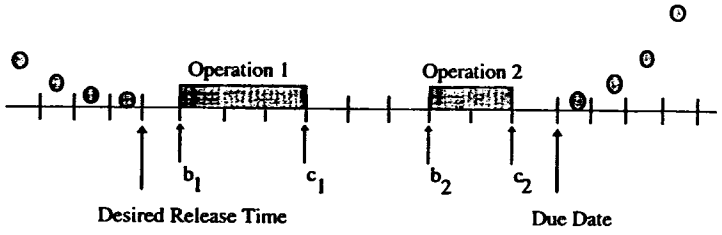

Figure 5. Part tardiness and earliness penalties.

The overall problem is to minimize the weighted tardiness and earliness penalties (9) subject to machine capacity, operation precedence, processing time, group, and run sequencing constraints (1)–(8). Key decision variables are operation beginning times $\{b_{ij}\}$, machine type to be used $\{h \in H_{ij}\}$, run beginning times $\{b^b_{hgn}\}$ and completion times $\{c^b_{hgn}\}$, and slack variables. Once these variables are determined, other variables can be easily derived. Among the constraints, machine capacity and group constraints are "coupling constraints" as they couple together decision variables associated with different parts and groups. Since the objective function and all coupling constraints 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.

4. Solution methodolgy

Similar to the pricing concept of a market economy, the Lagrangian relaxation method replaces "hard" coupling constraints (e.g., machine and buffer capacity constraints) by the payment of certain "prices" (i.e., Lagrange multipliers) for the use of a machine or a buffer at each enumeration step. The original problem can thus be decomposed into 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 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. Simple heuristics are then 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.

The method has been successfully used in the scheduling of identical machines and standard job shops [1]. The special features of the system considered here, however, complicate the formation and resolution of the subproblems. Complications are caused by the existence of both enumeration and resolution increments, group-dependent setups, and unspecified buffering and run processing times. As opposed to [1], where machine capacity and precedence constraints are relaxed to form operation subproblems, only machine capacity and group constraints are relaxed in this study to form part and group subproblems. The precedence constraints are not relaxed to have better convergence and to obtain higher dual costs following [2,25]. These part-level and group-level subproblems are solved by using dynamic programming (DP).

4.1. *The Lagrangian relaxation framework*

Machine capacity and group constraints (1)–(3) are first "relaxed" by using Lagrange multipliers $\{\pi_{kh}\}$ and $\{\gamma_{gkh}\}$, respectively, and the Lagrangian is formed as:

$$L \equiv \sum_i \left[w_i T_i^2 + \beta_i E_i^2\right] + \sum_{k,h \in H_s} \left\{\pi_{kh}\left[\sum_{i,j}\left[v_{ij}\delta_{ijkh}\right] + m_{kh} - M_{kh}\right]\right\}$$

$$+ \sum_{k,h \in H_s} \left\{\pi_{kh}\left[\sum_{g,n} \varphi_{khgn} + \phi_{kh} - M_{kh}\right]\right\}$$

$$+ \sum_{g,k,h \in H_s} \left\{ \gamma_{gkh} \left[\sum_{i,j} \delta_{ijkh} + \sum_{n} \zeta_{khgn} + \sigma_{khg} - \sum_{n} \varphi_{khgn} \right] \right\}. \tag{10}$$

With the multipliers given, the "relaxed problem" is to choose key decision variables to minimize the Lagrangian L subject to precedence constraints, arrival time constraints, and processing time requirements (4) – (8). After regrouping relevant terms within L, the problem is decomposed into three classes of subproblems, including part subproblems, group subproblems, and a subproblem for slack variables.

In the following subsections, the above subproblems and their resolutions will be presented. The selection of a route from a set of alternatives routes, the updating of multipliers, and the development of heuristics will also be discussed.

4.2. *Part subproblems and their resolutions*

Collecting all the terms in (10) related to tank i leads to:

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

with

$$L_{ij}(b_{ij}, h) \equiv v_{ij} \sum_{t=b_{ij}}^{b_{ij}+t_{ijh}-1} \tilde{\pi}_{th}, \qquad \text{if } h \notin H_s, \text{ and}$$

$$L_{ij}(b_{ij}, h) \equiv \sum_{t=b_{ij}}^{b_{ij}+t_{ijh}-1} \tilde{\gamma}_{gth}, \qquad \text{if } h \in H_s, \tag{11}$$

subject to operation precedence constraints (6) and the arrival time constraint (7). In the above, symbols $\tilde{\pi}_{th}$ and $\tilde{\gamma}_{gth}$ are introduced to simplify the presentation, with $\tilde{\pi}_{th} \equiv \pi_{kh}/R$ and $\tilde{\gamma}_{gth} \equiv \gamma_{gkh}/R$, where $k \equiv \lfloor t/R \rfloor$. The decision variables are operation beginning times $\{b_{ij}\}$ and the machine type to be used for each operation of tank i.

Since multiplier $\tilde{\pi}_{th}$ is the price for using a type h machine at time t, and $\tilde{\gamma}_{gth}$ the price for occupying a slot of a group g run on machine type h at time t, the cost $L_{ij}(b_{ij}, h_{ij})$ is related to machine utilization. A part subproblem thus reflects the balance between machine utilization cost and tardiness and earliness penalties.

As mentioned, DP is used for part subproblem solving. For clarity of presentation, the DP process for a part subproblem without buffering operations and time step reduction (i.e., $R = 1$) will be presented first. The consideration of time step reduction will then be briefly discussed.

4.2.1. *Dynamic programming for part subproblems*

DP can be implemented in the forward form as in [2, 25] or in the backward form. Since backward DP can be extended to deal with uncertainties, backward DP is used

in this study to solve part subproblems (11) for possible future extensions. DP stages correspond to individual manufacturing operations, and at each stage, the states are possible operation beginning times. The DP algorithm starts with the last operation (stage) having the following terminal cost:

$$V_{i,J_i-1}\left(b_{i,J_i-1}, h_{i,J_i-1}\right) \equiv w_i T_i^2 + L_{i,J_i-1}\left(b_{i,J_i-1}, h_{i,J_i-1}\right). \tag{12}$$

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 (6) and arrival time constraint (7):

$$\begin{aligned} V_{ij}(b_{ij}, h_{ij}) &\equiv \min_{\{b_{il}, h_{il},\, l \in I(i,j)\}} \{L_{ij}(b_{ij}, h_{ij}) + V_{il}(b_{il}, h_{il})\} \\ &= L_{ij}(b_{ij}, h_{ij}) + \min_{\{b_{il}, h_{il},\, l \in I(i,j)\}} V_{il}(b_{il}, h_{il}), \quad 1 \le j < J_i - 1. \end{aligned} \tag{13}$$

The second term on the right-hand side of (13) is to select the minimum cost from all possible beginning times $\{b_{il}\}$ for eligible machine types $\{h_{il}\}$ among possible alternative processing routes $I(i, j)$, and this minimization is subject to precedence constraints. The costs associated with individual beginning times, machine types, and routes are computed and compared, and the one with the smallest cost is selected.

The equation for the first stage is given by

$$V_{i0}(b_{i0}, h_{i0}) \equiv \beta_i E_i^2 + L_{i0}(b_{i0}, h_{i0}) + \min_{\{b_{il}, h_{il},\, l \in I(i,0)\}} V_{il}(b_{il}, h_{il}). \tag{14}$$

The optimal L_i^* is then obtained as the minimal cumulative cost at the first stage, i.e.,

$$L_i^* \equiv \min_{\{b_{i0}, h_{i0}\}} V_{i0}(b_{i0}, h_{i0}). \tag{15}$$

Finally, the optimal beginning times and the corresponding machine types can be obtained by tracing forward along the stages. The computational complexity for the above DP is $O(\sum_j |H_{ij}| K)$, as shown in [2], where $|H_{ij}|$ is the cardinality of H_{ij}.

The above DP procedure can be directly applied when time step reduction is used by considering all possible beginning times in each resolution increment. The characteristics of the piecewise linear stagewise cost (11) and the resulting piecewise linear cumulative cost, however, can be exploited for efficient subproblem resolution. It is straightforward that only vertices (resolution increments) of a piecewise linear cumulative cost function need to be computed and compared to find the minimum cumulative cost. The computational requirements, therefore, can be reduced by examining only those possible vertices [24].

4.2.2. Treatment of buffering operations

As mentioned, the "processing time" for a buffering operation is unspecified, the stagewise cost for a buffering operation thus cannot be computed as for a manufacturing

operation. Since the beginning time of a buffering operation equals its preceding operation's completion time, and the completion time equals its subsequent operation's beginning time, the following equality holds for a buffering operation (i, j) on buffer h:

$$L_{ij}(b_{ij}, h_{ij}) = v_{ij} \sum_{t=b_{ij}}^{c_{ij}} \tilde{\pi}_{th} = v_{ij} \sum_{t=c_{il_j}+1}^{b_{il}-1} \tilde{\pi}_{th} = v_{ij} \sum_{t=0}^{b_{il}-1} \tilde{\pi}_{th} - v_{ij} \sum_{t=0}^{t=c_{il_j}} \tilde{\pi}_{th}, \qquad (16)$$

where $(i, l) \in I_{ij}$ and $(i, j) \in I_{il_j}$. The cost for a buffering operation can thus be separated into two portions,

$$\left(v_{ij} \sum_{t=0}^{t=c_{il_j}} \tilde{\pi}_{th} \right),$$

to be merged with its preceding stagewise costs, and

$$\left(v_{ij} \sum_{t=0}^{b_{il}-1} \tilde{\pi}_{th} \right),$$

to be merged with its subsequent stagewise cost. In this way, backward DP as presented in the above can be used without explicit consideration of buffering operations.

4.3. *Group subproblems and their resolutions*

Collecting all the terms in (10) related to group g on machine h leads to:

$$\min_{\{b^r_{hgn}, c^r_{hgn}\}} L_{hg}, \quad \text{with } L_{hg} \equiv \sum_{n=1}^{N_{hg}} \sum_{t=b^r_{hgn}}^{c^r_{hgn}} \tilde{\pi}_{th} - \sum_{n=1}^{N_{hg}} \sum_{t=b^r_{hgn}+S^r_{hg}}^{c^r_{hgn}} \tilde{\gamma}_{gth}, \qquad (17)$$

subject to run sequencing constraints (5). The decision variables are run beginning times $\{b^r_{hgn}\}$ and completion times $\{c^r_{hgn}\}$ for all runs of group g on machine h. From (17), a group subproblem reflects the balance between the costs for using the setup machines and the value for hosting operations and/or runs.

Since the run processing time is unspecified, the beginning and completion times are all decision variables. The following equality has been employed to separate the cost associated with a run into two portions, one depending on the run beginning time, and the other on the run completion time:

$$\sum_{t=b^r_{hgn}}^{c^r_{hgn}} \tilde{\pi}_{th} - \sum_{t=b^r_{hgn}+S^r_{hg}}^{c^r_{hgn}} \tilde{\gamma}_{gth} = \left\{ \sum_{t=0}^{c^r_{hgn}} \tilde{\pi}_{th} - \sum_{t=0}^{c^r_{hgn}} \tilde{\gamma}_{gth} \right\} - \left\{ \sum_{t=0}^{b^r_{hgn}-1} \tilde{\pi}_{th} - \sum_{t=0}^{b^r_{hgn}+S^r_{hg}-1} \tilde{\gamma}_{gth} \right\}. \qquad (18)$$

Backward DP is used to solve the group subproblem similar to what was presented in the previous subsection. Since run beginning and completion times are all decision

variables, each run has two DP stages, one corresponding to run beginning time and the other to run completion time. The stagewise costs for a run beginning time stage and a run completion time stage are given by

$$\left\{ -\sum_{t=0}^{b^r_{hgn}-1} \tilde{\pi}_{th} + \sum_{t=0}^{b^r_{hgn}+S^r_{hg}-1} \tilde{\gamma}_{gth} \right\} \quad \text{and} \quad \left\{ \sum_{t=0}^{c^r_{hgn}} \tilde{\pi}_{th} - \sum_{t=0}^{c^r_{hgn}} \tilde{\gamma}_{gth} \right\},$$

respectively. With these stagewise costs, the backward DP procedure presented in subsection 4.2.1 can be used to solve group subproblems (17).

4.4. Slack subproblem

Collecting all the terms in (10) related to slack variables leads to

$$L_s \equiv \sum_{k,h} \pi_{kh} m_{kh} + \sum_{g,k,h \in H_g} \gamma_{gkh} \sigma_{khg}. \tag{19}$$

The solution to the above subproblem is

$$m_{kh} = \begin{cases} M_{kh}, & \text{if } \pi_{kh} < 0, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$\sigma_{khg} = \begin{cases} N_{hg}, & \text{if } \gamma_{gkh} < 0, \\ 0, & \text{otherwise.} \end{cases}$$

4.5. Dual problem and Lagrange multiplier updating

4.5.1. The dual problem

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

$$\max_{\{\pi_{kh}, \gamma_{gkh}\}} D, \quad \text{with} \quad D \equiv \sum_i L_i^* + \sum_{h,g} L_{hg}^* + L_s^* - \sum_{k,h} \pi_{kh} M_{kh}. \tag{20}$$

4.5.2. Updating Lagrange multiplier

The dual function D above is polyhedral concave, and the subgradient method is commonly used to solve this type of problems. The method, however, suffers from slow convergence, as shown in [19].

The bundle method [20,21] overcomes the slow convergence of the subgradient method by accumulating subgradients of points in a neighborhood of the current

iterate and storing them in a "bundle". A computationally-expensive step in the bundle method is to find a trial direction by solving a quadratic programming problem. The recently developed Reduced Complexity Bundle Method (RCBM) [23] has reduced the computational requirements in computing a trial direction by performing a projection of a bundle element onto a linear subspace. The RCBM is used to update the multipliers here.

4.5.3. Impact of time step reduction

Assume that K is the scheduling horizon, H the number of standard machine types, H_s the number of machines with group-dependent setups, and G the maximum number of groups per machine for machines in H_s. Rough estimates of the number of multipliers π_{kh} and γ_{gkh} are $K * (H + H_s)$ and $K * H_s * G$, respectively.

For a given scheduling problem, the number of multipliers can be controlled by the selection of R, the number of resolution increments within an enumeration step. When R increases, K is decreased, the number of multipliers is decreased, and less computational effort is needed for solving the high-level problem. In addition, time step reduction also reduces the computational requirements for subproblem solving, as mentioned in subsection 4.2.1. However, since the "prices" for individual resolution increments within one enumeration step are forced to be the same, these prices may not reflect the actual demand on machines. This inaccuracy in modeling may affect scheduling quality as will be seen in Test Case 4 of section 5.

4.6. Heuristics

The updating of multipliers is stopped after a fixed computation time, or a fixed number of iterations have been executed. Since machine capacity and group constraints have been relaxed, solutions of subproblems, when put together, generally do not constitute a feasible schedule. A heuristic procedure is used to adjust subproblem solutions to form a feasible schedule following [1]. A list of operations is first created by arranging all operations in the ascending order of their beginning times. Operations are then scheduled on their assigned machines as machines become available. If machine capacity constraints are violated at time t, a greedy heuristic based on the incremental change in J determines which operation should begin at that time slot, and which ones should be delayed.

Setup is determined based on a machine's status. If a machine has been set up for a group and the next tank scheduled belongs to that group, then no setup is needed. Otherwise a setup is needed. This rule also applies to the machine's initial status with the requirement of processing at least one tank. A buffering operation ends when the successor operation begins. If the successor operation can begin at the same time as the buffering operation begins, the buffering operation is skipped. Finally, if the buffer is full and all the subsequent machine types are busy, the tank will stay on and tie up the current machine until the buffer or one of the subsequent machine types is available.

This situation is slightly inconsistent with the model, where a buffering operation begins right after its preceding manufacturing operation.

The quality of a schedule obtained is quantitatively evaluated by its relative duality gap, which is the relative difference between the schedule cost J and the dual value D, i.e., Duality Gap $\equiv (J - D)/D \times 100\%$. To monitor the performance of the optimization process, the heuristic procedure is performed periodically after a certain number of iterations have been executed. The updating of multipliers is also stopped when the duality gap is within an acceptable range.

4.7. *Initialization of multipliers during schedule reconfiguration*

Since the above problem formulation and solution methodology are based on a "snapshot", i.e., the status of the shop at a particular time instant, scheduling is required to be performed either periodically or after the occurrence of major changes such as the arrivals of new orders, the breakdown of machines, etc. Rescheduling is initialized by using multipliers obtained for the previous schedule. As mentioned, the values of multipliers reflect the demand on machines. Since the status of the shop may not change significantly from one scheduling instant to the next, the multipliers may not change drastically. As will be seen in Test Case 6 of section 5, this multiplier initialization provides a better starting point for the optimization process, and significantly reduces the computational requirements.

5. Testing results

The method has been implemented using the object-oriented programming language C++, and extensive testing has been performed on a Sun Sparc10 workstation. Six test cases are reported here to demonstrate the performance of the method developed. For all test cases, the resolution time unit is one hour, and the multipliers are initialized at zero unless specified otherwise. The enumeration step equals the resolution time unit (i.e., $R = 1$) in the first three cases.

Case 1. This case is to demonstrate the effects of explicit modeling machines with group-dependent setups. There are two standard machine types, each with a single machine, one machine with group-dependent setups, and a buffer with capacity of one part in front of the machine with setups. Five parts, each with due date 0 and weight 1, are to be scheduled as summarized in table 1. Since all parts are due at time zero, there are no earliness penalties. All machines are available throughout the time horizon of 35 hours, and there are 245 multipliers.

The feasible schedule, Schedule 1.1, has a cost of 1647 with a relative duality gap 4.35%, and is obtained in 2.16 CPU seconds. The resulting operation beginning times are given in table 1, and the Gantt chart is shown in figure 6. It can be seen that tanks with the same setup requirements are grouped together to avoid excessive setups. It turns out that the schedule is optimal as verified by exhaustive search.

Table 1

Data and numerical results for Case 1.

Part i	Op. J	Mach. h	t_{ijh}	Group g	S^b_{hg}	Schedule (b_{ij}) 1.1	1.2	1.3
0	0	0	2			9	5	0
	1	1	–			11	7	–
	2	2	1	1	4	14*	10*	2*
	3	3	1			19	15	7
1	0	0	2			0	0	5
	1	1	–			–	–	7
	2	2	3	2	2	2*	2*	10*
	3	3	3			7	7	15
2	0	0	4			2	7	7
	1	1	–			6	11	11
	2	2	4	2	2	7	15*	15
	3	3	4			11	21	19
3	0	0	7			11	11	11
	1	1	–			18	18	18
	2	2	5	1	4	19	21*	19*
	3	3	5			24	30	28
4	0	0	3			6	2	2
	1	1	–			9	5	5
	2	2	2	3	1	11*	7*	7*
	3	3	1			15	10	10

*Setup is performed before the operation

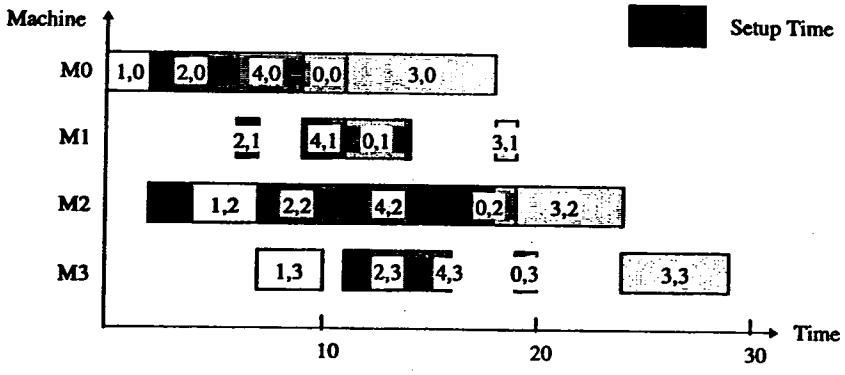

Figure 6. Gantt chart of Schedule 1.1, where tanks with the same setup requirements are grouped together.

To illustrate the advantage of explicit modeling of setups, the problem is resolved by treating the machine with setups as a standard machine type, with setup times lumped with operation processing times. The feasible schedule with respect to this simplified model, Schedule 1.2, is also shown in table 1, and has a cost of 2138 with a duality gap 0.24%. Compared to Schedule 1.1, this schedule requires a setup for each operation.

The problem is also solved by using a mixed modeling method, i.e., the optimization treats all machines as standard as discussed in the above, while the heuristics consider group dependent setups explicitly as described in subsection 4.6. The feasible schedule, Schedule 1.3, is also shown in table 1, and has a cost of 1946. This schedule is better than Schedule 1.2, but still requires an extra setup and has a larger cost as compared to Schedule 1.1. This mixed approach does have some advantages, e.g., simpler modeling, reduced number of constraints, and simpler dual problems as compared to the explicit modeling of setups. It may not, however, result in good feasible schedules since the optimization and the heuristics are, in fact, solving different problems. For the same reason, the duality gap may not provide useful information.

An indirect way to evaluate the work-in-process inventory is to examine the average percentage time that tanks are actually being processed, i.e.,

$$\mu \equiv \left[\left(\sum_{i=0}^{I-1} \frac{\text{Value added time for part } (i)}{\text{Total time in the system } (i)} \right) / I \right] \times 100\%.$$

where

$$\text{Value added time for part } (i) \equiv \sum_{j=0}^{J_i-1} t_{ijh^*},$$

and

$$\text{Total time in the system } (i) \equiv c_{i,J_i-1} - b_{i0} + 1.$$

In the above, h^* represents the machine type selected for operation (i, j), and buffering operation times are not counted in the value added time. The value of μ is thus the average percentage of value added time. From the work-in-process inventory viewpoint, the value of μ reflects the WIP inventory level, i.e., a larger value of μ corresponds to a lower WIP inventory level. The values of μ for Schedules 1.1, 1.2 and 1.3 are 72.62%, 57.04%, and 66.10%, respectively. The WIP inventory level of Schedule 1.1 is therefore much lower than that of either Schedule 1.2 or Schedule 1.3.

Case 2. This case is to show that better schedules can be generated by explicit modeling finite buffers. There are two standard machine types, each with a single machine, and a buffer with one-part capacity. All machines are available throughout the time horizon of 30 hours except that machine type 2 (M2) is not available from time period 8 to 11. Five parts are to be scheduled as summarized in table 2, where operations $(i, 1)$ are buffering operations. The number of multipliers equals 90.

Table 2

Data and numerical results for Case 2.

Part i	Op. J	Mach. h	t_{ijh}	d_i	w_i	Schedule 2.1 (b_{ij})	Schedule 2.2 (b_{ij})*
0	0	0	2			0	0 (0)
	1	1	–			–	–
	2	2	6	0	3	2	2 (2)
1	0	0	4			12	5 (5)
	1	1	–			–	– (12)
	2	2	2	3	1	16	16 (16)
2	0	0	3			16	9 (12)
	1	1	–			19	– (16)
	2	2	3	3	1	22	18 (18)
3	0	0	7			5	12 (15)
	1	1	–			12	–
	2	2	4	3	1	18	21 (22)
4	0	0	3			2	2 (2)
	1	1	–			5	–
	2	2	4	0	2	12	12 (12)

* The adjusted schedule for Schedule 2.2 is shown in the parentheses.

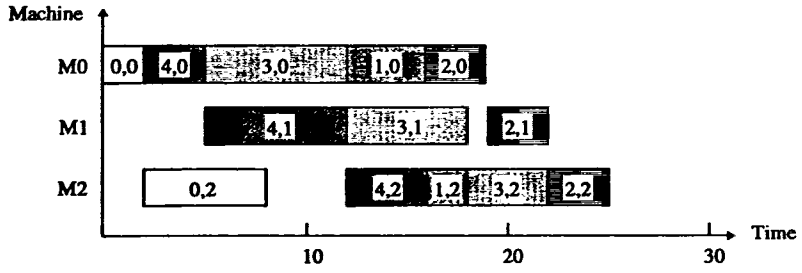

Figure 7. Gantt chart of Schedule 1.2, where the finite buffer is appropriately utilized.

The feasible schedule, Schedule 2.1, has a cost of 1558 with a duality gap 0.21%, and is obtained within one second. The resulting operation beginning times are summarized in table 2, and the Gannt chart is shown in figure 7. It can be seen that the buffer is appropriately utilized to obtain the feasible schedule.

For comparison, the problem is re-solved by ignoring "buffering operations". The schedule, Schedule 2.2, without considering the finite buffer has a cost of 1523 with a duality gap 0.16%. The operation beginning times are also summarized in table 2. The Gantt chart is shown in figure 8, where operations on M1 (buffer) are inserted whenever needed. It can be seen that the schedule is not "really" feasible with respect

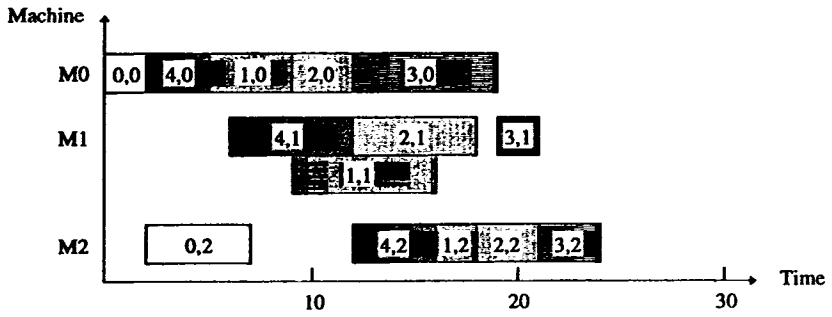

Figure 8. Gantt chart of Schedule 2.2, where the finite buffer constraint is violated.

to the original problem, e.g., part 1 and part 4 cannot be stored in the buffer at the same time periods from 9 to 12, etc. By simply prolonging the time a part spent on machines until the buffer or the next machine becomes available, a feasible schedule can be obtained. This schedule has a cost of 1566, which is greater than the cost of Schedule 2.1. The values of μ for Schedules 2.1 and 2.2 are 76.27% and 72.56%, respectively. The WIP inventory level of Schedule 2.1 is therefore lower than that of Schedule 2.2.

Case 3. This case is to show that parts with fork-join process plans can be effectively managed. There are five standard machine types each with a single machine. Four parts, each with due date 0, weight 1, and a routing as shown in figure 9, are to be scheduled. The detailed operation processing times are summarized in table 3. All machines are available throughout the time horizon of 50 hours, and there are 250 multipliers.

The feasible schedule, Schedule 3.1, has a cost of 2968 with a duality gap 3.42%, and is obtained in 1.56 seconds. The resulting operation beginning times are given in table 3, and the Gantt chart is shown in figure 10. It can be seen that part 0 and part 2 select route 1, and part 1 and part 3 select route 2.

If routing is to be selected manually, then there are 16 (2^4) routing possibilities. Among them, six have "balanced" load between the routes – two parts per route. The schedules for these 6 routing selections are individually generated. It turns out that the best of the 6 schedules is exactly the same as Schedule 3.1, and the worst has a cost of 3413, compared to 2968 for Schedule 3.1. For more complicated problems, it will be difficult to enumerate and solve all routing possibilities. The method presented here embeds route selection within the DP process, and efficiently overcomes this combinatorial difficulty.

The following three test cases draw data from Toshiba's HamaKawasaki Works, which produces GISs. In these test cases, all machines are available throughout the time horizon, and parts to be scheduled may have different due dates and weights.

Table 3

Data and numerical results for Case 3.

Part i	Op. j	Mach. h	t_{ijh}	Schedule 3.1(b_{ij})
0	0	0	4	2
	1	1	8	6
	2	2	6	–
	3	3	4	–
	4	4	5	14
1	0	0	5	6
	1	1	10	–
	2	2	7	11
	3	3	5	18
	4	4	8	23
2	0	0	7	11
	1	1	12	18
	2	2	10	–
	3	3	8	–
	4	4	10	31
3	0	0	2	0
	1	1	7	–
	2	2	4	2
	3	3	4	6
	4	4	3	10

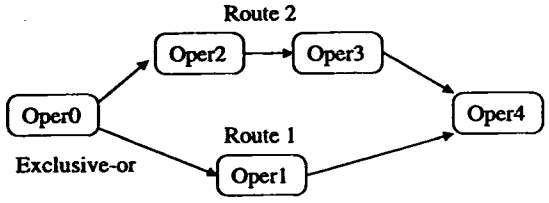

Figure 9. The routing for case 3.

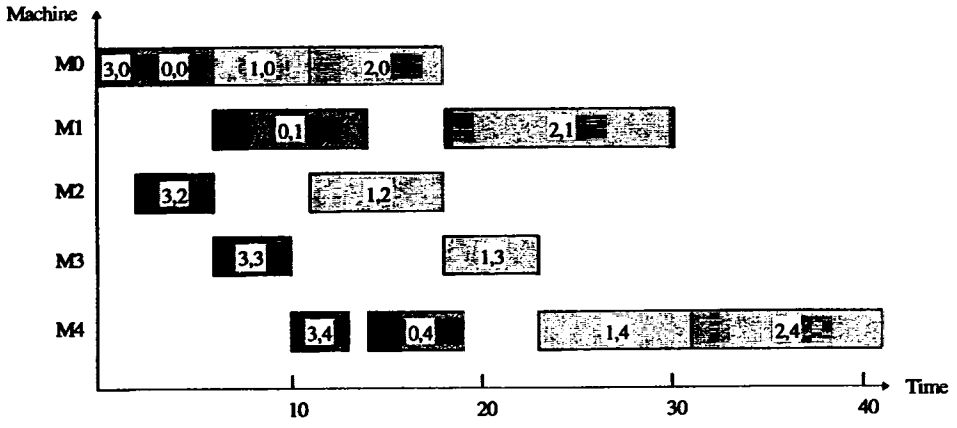

Figure 10. Gantt chart of Schedule 3.1, where "balanced" routing selections can be observed.

Table 4

Description of data for Cases 4 and 5.

Case no.	No. of parts	Ave. no. of operations/parts	No. of machines/ machine types	No. of primal decisions	Time horizon
4	40	12.95	45 / 19	518	290
5	80	14.10	47 / 21	1128	540

Cases 4 and 5. Table 4 summarizes the test data for Cases 4 and 5. Different R values, i.e., numbers of resolution increments within an enumeration, have been tested for Case 4 to illustrate the effects of time step reduction. Two stopping conditions for the optimization process are examined:

(C1) Time-dependent stopping criterion: the process is terminated when the number of dual function evaluations reaches 1000.

(C2) Quality-dependent stopping criterion: the process is terminated when the duality gap is less than 5%.

Table 5

Numerical results for Case 4.

		C1: 1000 dual function evaluations			C2: duality gap is less than 5%		
R (hrs)	Number of multipliers	Feasible cost	Duality gap	CPU time (min:sec)	Feasible cost	Duality gap	CPU time (min:sec)
1	5510	318155	16.15%	6:08	318075	4.76%	18:29
5	1102	12703	10.21%	4:01	12204	4.45%	7:59
10	551	3204	5.52%	2:15	3204	4.49%	2:23
20	285	880	16.67%	1:16	880	4.46%	2:37
30	190	423	58.63%	1:10	–	–	–

– : Cannot obtain a result with duality gap less than 5% for more than 10 minutes.

From the testing results summarized in table 5, it can be seen that the number of multipliers increases as R decreases, and longer computation time is needed for a given number of dual function evaluations with the increase of the number of multipliers. When no time step reduction is used (i.e., $R = 1$ hour), good schedules usually can be generated but with long computation times. When the enumeration step is large (e.g., $R = 30$), the time spent for a given number of dual function evaluations decreases significantly. The duality gap, however, may be large. The reason, as mentioned in subsection 4.5, is that the inaccuracy of the model increases with the increase of R.

A suitable enumeration step is thus important for good performance. Considering the quality of the schedule and the time needed, $R = 10$ is appropriate for GIS production. This enumeration step is also used for subsequent test cases.

Case 5 is used to show quality improvement as the number of function evaluations increases. There are a total of 1134 multipliers, and numerical results are summarized in table 6. As the number of dual function evaluations increases, better dual costs are obtained at the sacrifice of computation times. Good feasible costs, however, usually can be obtained within a reasonable number of dual function evaluations.

Table 6

Numerical results for Case 5.

Number of dual function evaluations	Primal cost	Dual cost	Dualit gap	CPU time (min:sec)
500	31367	24127	30.01%	4:25
1000	31094	27784	11.91%	8:21
1500	31094	28944	7.42%	12:20
2000	31094	29454	5.57%	16:27

Case 6. This case is to show that better multiplier initialization can significantly reduce the computation time. According to the schedule generated in Case 5, 11 parts are assumed completed in the first three days (6 ten-hour shifts), and 15 new parts are assumed to have arrived. The schedule is reconfigured at the beginning of the fourth day, based on the procedure presented in subsection 4.7, with 84 parts for a total of 1204 operations to be scheduled. For comparison, the optimization process is also performed by initializing the multipliers at zero.

Table 7

Numerical results for Case 6.

Multipliers initialization	Number of dual function evaluations	Duality gap	CPU time (min:sec)
at 0	2000	6.47%	16:45
from previous process	500	4.43%	4:43

The numerical testing results are summarized in table 7. As expected, the CPU time for generating a high-quality schedule can be drastically decreased by using the multipliers from the previous process.

6. Concluding remarks

The modeling of the special GIS production features and the mathematical resolution of the resulting scheduling problem have been presented. The effective handling of group-dependent setups is of particular significance, and should shed light on many similar problems. The treating of finite capacity buffers as machines, the embedded routing selection mechanism, and the time-step reduction technique also contribute to the state-of-the-art and practice of scheduling methodology. Numerical results show that the method can generate high quality schedules in a timely fashion.

Appendix: A list of symbols

a_i : arrival time of raw materials for tank i;
b_i : desired release time of tank i;
b_{i0} : beginning time of the first operation of part i;
b_{ij} : beginning time of operation (i, j);
b^r_{hgn} : beginning time of run (h, g, n);
c_{i,J_i-1} : completion time of the last operation of part i;
c_{ij} : completion time of operation (i, j);
c^r_{hgn} : completion time of run (h, g, n);
d_i : due date of tank i;
E_i : earliness of tank i;
G_h : total number of groups on machine type h;
g : group index, $1 \le g \le G_h$;
H : set of standard machine types;
H_s : set of machine types with group-dependent setups;
H_{ij} : set of eligible machine types for operation (i, j) that belongs to the set H or H_s;
h : machine type index;
(h, g, n) : the nth run of group g on machine type h;
I : total number of tanks to be scheduled;
I_{ij} : set of immediate subsequent operations of (i, j);
i : tank index, $0 \le i \le I - 1$;
(i, j) : the jth operation of tank i, $0 \le j \le J_i - 1$;
J_i : total number of operations of tank i;
j : operation index;
K : time horizon in enumeration step;
k : enumeration step index;

M_{kh} : average number of h type machines available at enumeration step k;
m_{kh} : non-negative slack variable satisfying $0 \le m_{kh} \le M_{kh}$;
N_{hg} : total number of runs belonging to group g on machine type h;
n : run index, $1 \le n \le N_{hg}$;
R : number of resolution increments within one enumeration step;
S^r_{hg} : setup time of each run of group g on machine type h;
S_{ijl} : required "timeout" between operation (i, j) and (i, l);
T : time horizon in resolution increment;
T_i : tardiness of tank i;
t : time index in resolution increment, $0 \le t < T$;
t^r_{hgn} : processing time of run (h, g, n);
t_{ijh} : processing time of operation (i, j) on machine type h;
v_{ij} : tank volume for buffers;
w_i : weight of tardiness penalty for tank i;
x : the xth resolution increment within an enumeration step, $0 \le x < R$;
β_i : weight of earliness penalty for tank i;
δ_{ijkh} : fraction of the enumeration step k that operation (i, j) is active on machine type h;
ϕ_{kh} : non-negative slack variable satisfying $0 \le \phi_{kh} \le M_{kh}$;
γ_{gkh} : Lagrange multiplier of group g on machine type h at enumeration step k;
φ_{khgn} : fraction of an enumeration step k that run (h, g, n) is active on machine type h;
π_{kh} : Lagrange multiplier of machine type h at enumeration step k;
σ_{khg} : non-negative slack variable satisfying $0 \le \sigma_{khg} \le N_{hg}$;
ξ_{khgn} : fraction of enumeration step k that run (h, g, n) is being set up on machine type h.

Acknowledgements

This work was supported in part by Toshiba Corporation for the first author's sabbatical leave, by the National Science Foundation Grants DDM-9119074 and DMI-9500037, and the Advanced Technology Center for Precision Manufacturing at the University of Connecticut. The authors would like to thank Dr. Shinsuke Tamura and Dr. Sadakazu Watanabe of Systems & Software Engineering Laboratory, Mr. Satoshi Ohyama, Mr. Takashi Miyatani, Mr. Akira Yamgishi, and Mr. Hiroyuki Yamada of HamaKawasaki Works of Toshiba for their valuable support and contributions to this project. The authors are grateful to the referees for their helpful comments.

References

[1] P.B. Luh and D.J. Hoitomt, Scheduling of manufacturing systems using the Lagrangian relaxation technique, IEEE Transactions on Automatic Control 38(1993)1066.

[2] H. Chen, C. Chu, and J.M. Proth, A more effective Lagrangian relaxation aproach, *Proc. IEEE Int. Conf. on Robotics and Automation*, Nagoya, Japan, 1995, p. 496.

[3] L.J. Krajewski, B.E. King, L.P. Ritzman and D.S. Wong, Kapan, MRP and shaping the manufacturing environment, Management Science 33(1987)39.

[4] S.S. Panwalkar, R.A. Dudek and M.L. Smith, Sequencing research and the industrial scheduling problem, *Symposium on the Theory of Scheduling and Its Applications*, 1973, p. 29.

[5] K.R. Baker, *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

[6] W.C. Driscoll and H. Emmons, Scheduling production on one machine with changeover costs, AIIE Trans. 9(1977)338.

[7] C.H. White and R.C. Wilson, Sequence dependent set-up times and job sequencing, Int. J. Prod. Res. 15(1977)191.

[8] I. Adiri and N. Amit, Route-dependent open shop scheduling, IIE Trans. 15(1983)231.

[9] S.K. Gupta, *N* jobs and *M* machines job-shop problems with sequence dependent setup times, Int. J. Prod. Res. 20(1982)643.

[10] B.L. Golden and W.R. Stewart, Empirical analysis of heuristics, in: *The Traveling Salesman Problem*, E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan and D.B. Shmoys, eds., Wiley, Chichester, 1985, p. 361.

[11] E.Uskup and S.B. Smith, A branch and bound algorithm for two-stage production sequencing problems, Operations Research 23(1975)118.

[12] I.M. Ovacik and R. Uzsoy, Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup times, Int. J. Prod. Res. 33(1995)3173.

[13] C. Zhou and P.J. Egbelu, Scheduling in a manufacturing shop with sequence-dependent setups, Robotics and Computer-Integrated Manufacturing 5(1989)73.

[14] S.C. Kim and P.M. Bobrowski, Impact of sequence-dependent setup time on job shop scheduling performance, Int. J. Prod. Res. 32(1994)1503.

[15] P.A. Rubin and G.L. Ragatz, Scheduling in a sequence dependent setup environment with generic search, Computers and Operations Research 22(1995)85.

[16] K. Mori, M. Tsukiyama, and T. Fukuda, A hybrid scheduling method for the job shop scheduling problem, Transactions of the Institute of Electrical Engineers of Japan, Part C, 9(1992)568.

[17] T.A. Owens and P.B. Luh, Improving the time resolution of schedules generated via Lagrangian relaxation algorithms, *Proc. 1992 IEEE Int. Conf. on Systems, Man and Cybernetics*, Chicago, IL, 1992, p. 1390.

[18] J. Wang and P.B. Luh, Scheduling of a machining center, Mathematical and Computer Modeling 24(1996)203.

[19] R.N. Tomastik and P.B. Luh, The facet ascending algorithm for integer programming problems, *Proc. 32nd IEEE Conf. on Decision and Control*, San Antonio, TX, 1993, p. 2880.

[20] C. Lemarechal, Bundle methods in nonsmooth optimization, in: *Nonsmooth Optimization, Proc. of an IIASA Workshop*, C. Lemarechal and R. Mifflin, eds., Pergamon Press, 1978.

[21] J.B. Hiriart-Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithms; I and II*, Springer, Berlin, 1993.

[22] R.N. Tomastik and P.B. Luh, A reduced-complexity bundle method for maximizing concave nonsmooth functions, *Proc. 35th IEEE Conf. on Decision and Control*, Kobe, Japan, Dec. 1996.

[23] L. Gou, Holonic manufacturing scheduling: Architecture, cooperation mechanism, and implementation, Ph.D. Dissertation, University of Connecticut, 1997.

[24] J. Wang and P.B. Luh, A combined Lagrangian relaxation and dynamic programming algorithm for job shop scheduling, *Proc. 5th Rensselaer Int. Conf. on Computer Integrated Manufacturing and Automation Technology*, Grenoble, France, May 1996, p. 3.