Technical Memorandum Number 818

Coordination of Outsourcing Operations

by

Tolga Aydinliyim
George L. Vairaktarakis

September 2007

Department of Operations
Weatherhead School of Management
Case Western Reserve University
330 Peter B Lewis Building
Cleveland, Ohio 44106

Coordination of Outsourcing Operations

Tolga Aydinliyim * George L. Vairaktarakis †

September 17, 2007

Abstract

A set of manufacturers outsource certain operations to a single third-party. Each available day of production at the third party can be booked at a given cost announced by him. Knowing these costs, manufacturers book available production days in a first-come-first-served (FCFS) order so as to optimize their individual cost. The cost for each manufacturer consists of booking and work-in-progress (WIP) costs as expressed by the weighted flow time. When window booking is completed, the third-party identifies a schedule that minimizes the total cost incurred by all manufacturers. This coordination reduces the total cost but may result to higher costs for a subset of manufacturers. For this reason, the third-party devises a savings sharing scheme with which the monetary benefit for each manufacturer is greater following the coordination. In this article we present algorithms for the problem considered, as well as savings sharing schemes that make coordination a better alternative for all parties involved. The highlight of our experiments is that the costs of the production chain can be reduced by an average of 32% if one-third of the members let the third-party cover their increased WIP cost in exchange for 40-55% of the total savings.

*Decision Sciences Department, Lundquist College of Business, 1208 University of Oregon, Eugene, OR 97403-1208, tolga@uoregon.edu

†Department of Operations, Weatherhead School of Management, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106-7235, gxv5@case.edu

1 Introduction and Literature Review

Today's business-to-business world is made up of continuously growing supply chains which create a complex business environment that is hard to control. Huge networks made up of competing suppliers, manufacturers and distributors create situations where individual decisions are affected by the decisions of other parties involved in the network. The quality of the decisions depends on the amount of information the decision makers have about other members of the supply chain. Today, many industry sectors have created portals where key information is shared amongst the members of the network. This allows the members of a supply chain to cooperate so as to reduce their individual costs as well as the cost of the entire supply chain. In this article, we model such cooperation and provide incentive schemes.

Specifically, we consider models whereby a group M of manufacturers book third-party ($3P$) capacity for their operations which require little or no set-up between them but have different processing time requirements (e.g. finishing operations, testing operations). The set-up times between the jobs that belong to different manufacturers are also insignificant. The $3P$ announces his capacity availability as well as the booking cost for each day of production, referred to as a *manufacturing window*. The booking price of each window might reflect peak demand periods or timeliness with respect to the start of the planning horizon. Knowing the availability and the costs of the manufacturing windows, manufacturers book $3P$ capacity independently in a first-come-first-serve (FCFS) basis with the objective to minimize their individual costs expressed as the total weighted flow time plus booking costs. Therefore, for each manufacturer there is a trade-off between booking expensive early windows and cheap late windows. When all bookings are finished the $3P$ offers a coordinated schedule by rescheduling all jobs as if the jobs from different manufacturers belong to a single party. Overall cost reductions are obvious and significant but some manufacturers might end up with higher individual costs. Therefore, the $3P$ provides incentives to each manufacturer so that everyone benefits from coordination. On the other hand, the $3P$ also creates a mechanism that generates benefits for himself via *booking refunds* and *re-bookings*. Specifically, some of the manufacturing windows are released in the coordinated schedule. The $3P$ refunds a fraction of the booking costs of these windows and keeps the rest for himself. Besides, he generates additional booking revenues by re-selling these windows.

In this model we assume complete information sharing amongst all manufacturers. This information sharing protocol is in fact becoming popular in industry. One such portal known as eHub, is in use at Cisco Systems. This is a private trading e-marketplace that provides a central point for planning and execution of tasks across the company's extended manufacturing supply chain (Grosvenor and Austin (2001)). Cisco's eHub creates opportunities to perform coordinated

planning and production among members of the network leading to dramatic cost and inventory reductions. By 2001 successful implementation of the idea resulted in inventory reductions of 45%, order cycle time reductions of 70% and increased productivity across the entire supply chain.

The model that we study in this paper finds applications in highly connected networks similar to that of Cisco's (see Figure 1). Orders from customers are stored in Cisco's enterprize resource planning (ERP) database and sent to the related manufacturing partners over the virtual private network (VPN). These manufacturing partners correspond to the group of manufacturers M in our model. Within Cisco's supply chain there are supply partners and other contractors who could see information on the network because their own production systems are also connected to Cisco's ERP system. Consider a supplier in this network that provides sub-assemblies to the aforementioned manufacturing partners. The type of supplier-manufacturer relationship whereby a manufacturer outsources operations to a single supplier is called *sole sourcing* in procurement literature. For a classification of such sourcing policies see Elmaghraby (2000). We consider a group of manufacturing partners who sole-sources from a single supplier. This supplier plays the role of the third-party $3P$ in our model. Similarly, one could think of a contractor in the same network who performs the testing operations for the assemblies or components produced by the manufacturing partners. This situation is equivalent to a model where manufacturing partners M outsource their testing operations to a single contractor in Cisco's network, who plays the role of the single third-party $3P$ in our model. Recall that the production schedules of all parties involved are transparent to everyone because they are all connected to Cisco's information sharing portal. Therefore, coordinated capacity and production planning opportunities exist within this framework. In our model, we consider such coordination possibilities and provide incentive schemes to make everyone benefit from coordination.

MyUMC[1], UMC's total online supply chain portal, is another example that demonstrates how a large Electronic Contract Manufacturer (ECM) provides information sharing and real-time capacity booking to her customers (i.e., the manufacturers who outsource). UMC is a world-leading semiconductor foundry, specializing in the contract manufacturing of customer designed integrated circuits for high performance semiconductor applications. Founded in 1980 as Taiwan's first semiconductor company, UMC currently employs over 12,000 people worldwide at its manufacturing factories and offices in Taiwan, Japan, Singapore, Europe, and the United States. To realize the vision of close collaboration with her customers as well as partners throughout the entire supply chain, UMC has developed and maintained MyUMC since 1998. This is a full-service information portal, offering customers 24-hour access to detailed account information such as manufacturing, engineering, design and financial data. In particular, MyUMC's capacity booking engine ATP

[1]http://my.umc.com

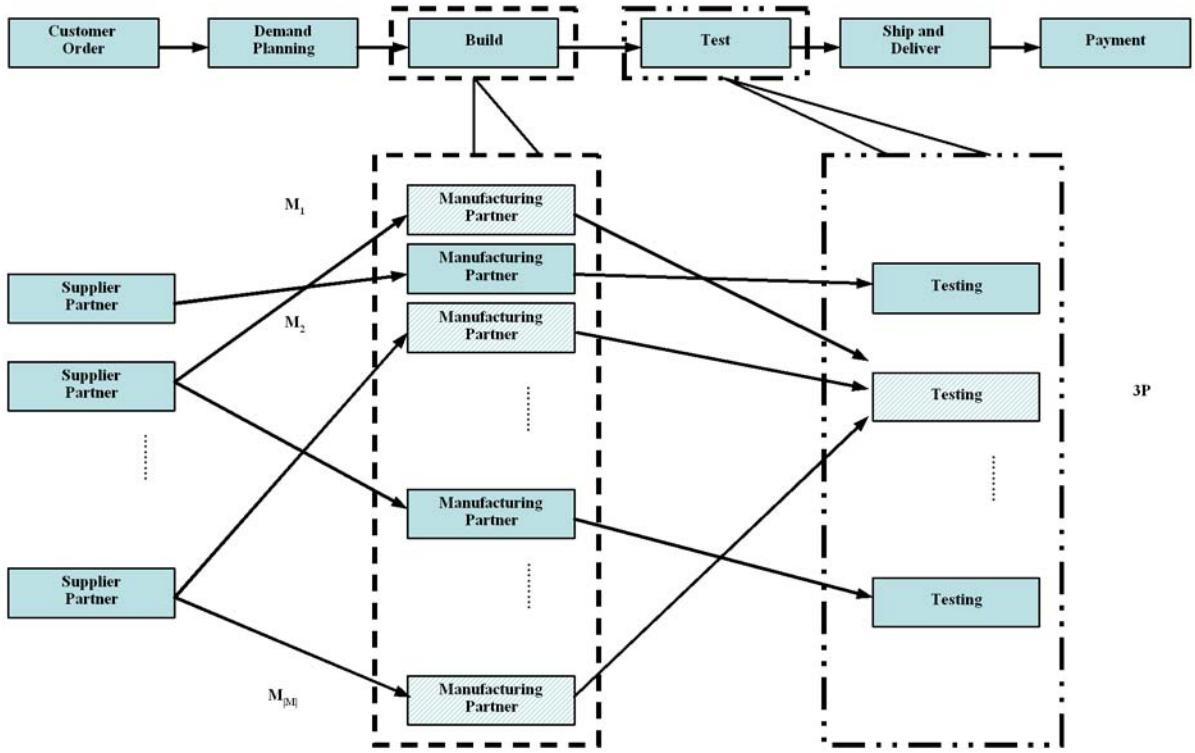

Figure 1: Order fulfillment process of the Cisco's networked supply chain and our model (Manufacturers M_1, M_2, ...,$M_{|M|}$ and the third-party $3P$)

(Available-to-Promise) allows customers to receive instant capacity availability information and to book production capacity in UMC's fabs online. Again, coordinated capacity and production planning opportunities exist within this framework.

The operational protocol of the Semiconductor Product Analysis and Design Enhancement (SPADE) Center[2] further illustrates more precisely our model. SPADE provides to local and nearby semiconductor companies such services as analysis and optimization of designs and products when their silicon prototypes are available in the form of silicon wafers or silicon dies. SPADE has a group of specialized facilities, including Focused Ion Beam, Emission Microscope, ESD Tester, Backside Preparation System (Chip UnZip), Laser Cutting System, etc., which can be booked at charges. SPADE announces the available booking periods for the following three weeks (similar to our finite number of manufacturing windows) and each manufacturer books capacity in a FCFS manner subject to already booked windows by earlier manufacturers. Rules of the charging scheme are available to customers[3], and include:

(a) *Jobs performed on Sundays and public holidays will be charged at 2 × basic rate;*

[2]http://www.ust.hk/spade
[3]http://www.ust.hk/spade/pricelist.html

(b) *Jobs performed during non-office hours will be charged at 1.5 × basic rate.*

The above rules show that there are different types of charging rates corresponding to the arbitrary booking prices in our model. SPADE has also built a mechanism to allow a user to preempt another by paying a higher premium as follows:

(c) *Jobs preempting other jobs on queue will be charged at 1.5 × basic rate.*

This reflects an attempt for reservation adjustments carried out at the third-party (e.g., SPADE). Instead of using price-based preemption, our proposed solution will perform more sophisticated coordination based on cooperative game theory, to allow all parties involved to be better off through proper cooperation.

Besides the above mentioned examples, our model finds applications in manufacturing and service industries such as scheduling the use of a single maintenance facility of a multi-divisional firm for the repair and maintenance operations of its divisions.

The issue of cooperation in our model can be addressed as a game amongst manufacturers who sequence their jobs over the windows booked when they entered the system. Sequencing games are at the interface of sequencing and scheduling of operations and cooperative game theory. In sequencing and scheduling a set of jobs has to be processed according to a schedule that minimizes a given cost objective. A group of players each owning a subset of jobs, share the same processing resources. Each player may optimize her individual objective function, or the players can coordinate their activities so as to maximize the group savings. In the latter case an incentive payment scheme is necessary to ensure that every group member cooperates.

A sequencing game involves solving two problems. A combinatorial optimization problem where a schedule is found for any given coalition of players, and a game theoretic problem of allocating the savings produced by coordination. The subset of allocations where savings are distributed to players in such a way that no subset of players can be better off by seceding from the rest of the players and acting solely on their own behalf is called the *core* of a game. A typical single machine sequencing game found in the literature assumes that there are $|N|$ players each with a single job, each occupying a specific position in the initial job schedule, and each having his own cost criterion.

Sequencing games were introduced by Curiel, Pederzoli and Tijs (1989). They considered the simplest case of a single machine with no restrictions on the jobs. The cost criterion is weighted flow-time and it is proved that the corresponding sequencing games are *convex*. For such games Shapley (1971) showed that a core allocation is guaranteed to exist. For arbitrary regular cost criteria and for a special class of games (referred to as $\sigma_0 - Component\ Additive\ Games$) Curiel, Potters, Rajendra Prasad, Tijs and Veltman (1994) proposed a core allocation known as the $\beta-rule$. Curiel, Hamers and Klijn (2002) presented a survey of all seminal papers since 1989 on sequencing games,

core allocations, and convexity issues. Literature on sequencing games where each player has more than one job to be processed is quite limited. Calleja, Estévez-Fernández, Borm and Hamers (2004) studied a single machine setting where each player might have more than one job to process and each job might be of interest to more than one players. They showed that the core of these games is non-empty only if the underlying cost functions are additive with respect to the initial order of jobs. Calleja, Borm, Hamers, Klijn and Slikker (2002) considered a two-machine problem where each player has precisely two operations, each processed on a dedicated machine. The cost criterion is the maximal weighted completion time of the two operations. It is shown that these games are not convex but a core allocation exists. Hamers, Borm and Tijs (1995) considered single machine games for jobs with release times and the weighted flow time cost criterion and showed that they are not convex except for instances with unit processing times or unit weights. Nevertheless, they proved that the core of these games is non-empty. Borm, Fiestra-Janeiro, Hamers, Sánchez and Voorneveld (2002) considered problems with job due dates. For three different due date related cost criteria they showed that the core is non-empty. Convexity was proven for only a special subclass. Hamers, Klijn and van Velzen (2002) imposed precedence constraints on the jobs and proved that the corresponding games are convex when the precedence network consists of parallel chains. Hamers, Klijn and Suijs (1999) considered games with m parallel identical machines and no restrictions on the jobs. The cost criterion is weighted flow time. Existence of core allocations is proved for $m = 2$ and for some special cases when $m > 2$. Recently, Cai and Vairaktarakis (2006) studied a production planning setting involving a single third party and multiple manufacturers each outsourcing a subset of jobs to the third party. The cost criterion consists of tardiness penalties plus booking costs that can take on one of two values reflecting peak and off-peak production. Despite non-convexity, core allocations are identified.

Literature at the interface of scheduling and cooperative game theory is quite limited. However, the issue of coordination where multiple parties compete for common resources has been studied in various fields such as economics, computer science and supply chain management. For example, *congestion games* were introduced by Rosenthal (1973), where it is shown that congestion games admit a potential function, therefore a Nash equilibrium exists. Congestion games can be used to model coordination and pricing issues in highways and communication networks (eg. Internet pricing). Sandholm (2002) considered the problem of ensuring the efficient use of a highway network and suggested pricing schemes. On the other hand, literature on load balancing in communication networks focuses on finding bounds on the ratio between the total cost of the uncoordinated equilibrium and the cost of the centralized solution. Papadimitriou (2001) used the term *price of anarchy* to denote the worst-case ratio between a Nash outcome and the social optimum. The *decentralization cost* is also commonly used in supply chain management literature. Another ap-

plication of decentalized scheduling in a factory environment is by Wellman and Walsh (2001). They investigated the existence and the quality of equilibrium prices and presented two auction mechanisms.

Our work is closely related to the cooperative sequencing games literature. In our model, we allow each manufacturer to schedule multiple jobs as in Calleja et al. (2004), and manufacturing costs include weighted flow time costs as in Curiel et al. (1989). Our main modeling contribution is the addition of a more realistic production planning setting where the third-party capacity is represented by non-contiguous manufacturing windows with limited capacity, and a composite objective function which is the sum of the work-in-process (WIP) costs plus the booking costs of the manufacturing windows. Our cooperative game is also novel

The rest of the article is organized as follows. We start the next section with a formal definition of our model and assumptions. In Section 3, we present three heuristics that provide near optimal solutions to our problem, and two different lower bounding schemes to assess the quality of the heuristics. In Section 4, we discuss coordination results and saving sharing schemes for a "fair" allocation of savings. We present the results of our computational experiments and report our insights in Section 5. Concluding remarks are made and future research directions are discussed in Section 6.

2 Model Formulation

Let M be a set of manufacturers who outsource their jobs, say $N = \{J_1, \ldots, J_n\}$ to a single third-party $3P$. The set of jobs that belong to manufacturer $m \in M$ is denoted by N_m. Each job has known processing time p_j and weight w_j, i.e. the unit work-in-process (WIP) cost of J_j. Party $3P$ has T available days of production (each referred to as a *manufacturing window*). The set of manufacturing windows is $\Gamma = \{W_1, \ldots, W_T\}$. Window W_k costs h_k to book as announced by $3P$ at time 0, for $1 \leq k \leq T$. The manufacturing windows are non-contiguous, i.e. there is a downtime G between two consecutive windows. Following the announcement of the available manufacturing windows and the associated booking costs, the manufacturers make reservations on a FCFS basis, say in the order $1, 2, 3, \ldots, |M|$.

Let ${\sigma_0}^m$ be an initial optimal schedule for manufacturer m and $\mathcal{W}_{{\sigma_0}^m}$ be the corresponding window collection. Manufacturer m jointly optimizes the schedule ${\sigma_0}^m$ of his jobs and the subset $\mathcal{W}_{{\sigma_0}^m} \subseteq \Gamma$ of windows selected. The joint optimization is made so as to minimize the total flow time cost $\sum_{J_j \in N_m} c_j({\sigma_0}^m)$ plus the total booking cost $\sum_{W_k \in \mathcal{W}_{{\sigma_0}^m}} h_k$. Every job is shipped as soon as its processing is completed; this protocol is referred to as *immediate shipments*. Hence, $c_j(\sigma) = w_j \cdot {C_j}^\sigma$ where ${C_j}^\sigma$ is the completion time of J_j in schedule σ and w_j is the unit WIP cost associated with

job J_j.

Following the determination of $\mathcal{W}_{{\sigma_0}^m}$ for each $m \in M$, $3P$ reschedules all the jobs in N as if they belong to one manufacturer so as to obtain the best schedule σ^* and associated collection $\mathcal{W}_{\sigma^*}$ of windows. Schedule σ^* minimizes the total cost over all manufacturers and hence it is no worse than the schedule σ_0 obtained by concatenating ${\sigma_0}^1, {\sigma_0}^2, {\sigma_0}^3, \ldots, {\sigma_0}^{|M|}$.

Every manufacturer $m \in M$ would agree to go along with σ^* only if his cost is no greater than that of ${\sigma_0}^m$. Therefore, $3P$ must find an allocation of the savings produced by σ^* such that every manufacturer is better off by coordination. Normally, σ^* is expected to utilize fewer windows than σ_0 because of better utilization of idle times across all manufacturers. Party $3P$ refunds a fraction ρ $(0 \leq \rho \leq 1)$ of the booking savings and keeps the rest for himself.

We now state the assumptions of our model. All jobs are assumed to be of similar nature (e.g. testing operations, maintenance operations, generic assembly and fabrication operations, etc.) and therefore the changeover time between jobs that belong to different manufacturers is assumed to be 0. All announcements by $3P$ about the booking costs and the reservations by manufacturers are made before the start of the first manufacturing window in the planning horizon (time 0). Also, all jobs are assumed to be available for processing at time 0. The FCFS ordering of the manufacturers implies that:

$$\mathcal{W}_{{\sigma_0}^m} \subseteq \{W_1, \ldots, W_T\} \setminus (\mathcal{W}_{{\sigma_0}^1} \cup \mathcal{W}_{{\sigma_0}^2} \cup \mathcal{W}_{{\sigma_0}^3} \cup \ldots \cup \mathcal{W}_{{\sigma_0}^{m-1}}) \tag{1}$$

We assume that the total processing requirements of jobs in N_m do not exceed the available processing hours of windows in $\{W_1, \ldots, W_T\} \setminus (\mathcal{W}_{{\sigma_0}^1} \cup \mathcal{W}_{{\sigma_0}^2} \cup \mathcal{W}_{{\sigma_0}^3} \cup \ldots \cup \mathcal{W}_{{\sigma_0}^{m-1}})$. Otherwise, manufacturer m could not be served by $3P$ and would seek another third party. Also, in case a job is preempted at the end of a manufacturing window, it resumes normally in the next booked window. Further, we assume that every manufacturing window $W_k \in \Gamma$ has precisely L hours of production for k = 1,...,T. The number of manufacturing windows required to process jobs in N is ω, where $\omega = \lceil \frac{\sum_{J_j \in N} p_j}{L} \rceil$.

Our final assumption relates to the information sharing structure for the parties involved. We assume that the third-party has complete information about the costs of the individual manufacturers while he does the rescheduling of all the jobs. This is reasonable in terms of the processing times as they are easily verifiable, but some questions may be raised about the unit weights. When the manufacturers are members of the same network (as in the case of Cisco), or the players are the divisions of the same parent company (as in the problem of scheduling the use of a single repair and maintenance facility by the divisions of a single firm) we may assume that the third-party has the precise cost information or that the players are willing to share their true cost parameters with the global scheduler.

When the manufacturers are independent players as in the SPADE example, we assume that the third-party has the means to approximately check the validity of the cost parameters declared by the players according to their individual bookings. For example, the third-party would have identified a priority seeking player who reports a higher WIP cost parameter, had this player skipped an expensive earlier window during his initial booking. However, it is still possible for the manufacturers to report inaccurate values seeking advantage in the coordinated schedule. Therefore, we believe there is research to be done in various directions including the study of different information structures with private information, and the analysis of truth-revealing mechanisms.

As described earlier, each manufacturer as well as $3P$ face the same optimization problem on a different set of available windows. For ease of presentation, we solve the centralized problem faced by $3P$ and assume that the manufacturers as well as subgroups of manufacturers adopt the same algorithms. Specifically, $3P$ schedules the jobs in N so as to minimize the total cost by utilizing windows in Γ.

3 Heuristics

In this section, we prove that the associated production planning problem is NP-Hard and develop algorithms to attain approximate solutions and lower bounds. Suppose π is a single machine schedule of jobs in N and ${F_j}^{\pi}$ the completion time of the j^{th} job in π. We want to partition π in ω parts and find the total weight of the completed jobs in each part. This is done as follows. Let $t_i = i \cdot L$, and define

$$w(t_i) = \sum_{\substack{J_j \in N \\ t_{i-1} < {F_j}^{\pi} \leq t_i}} w_j \,, \quad i = 1, \ldots, \omega. \tag{2}$$

Let B_i be the set of jobs that finish at time ${F_j}^{\pi}$: $t_{i-1} < {F_j}^{\pi} \leq t_i$. Therefore, given π we obtain B_1,B_2,B_3,...,B_ω. We introduce an indicator variable ${y_j}^i$ that shows whether J_j belongs to B_i; i.e., ${y_j}^i = 1$ if $t_{i-1} < {F_j}^{\pi} \leq t_i$, 0 otherwise. Another indicator variable ${z_i}^k$ take on value 1 if jobs in B_i utilize window W_k , 0 otherwise. Let D_k be the ending time of W_k. Then, the objective function is given by the following expression:

$$Z^* = \min_{\sigma} \sum_{W_k \in \Gamma} \sum_{i=1}^{\omega} \sum_{J_j \in N} w_j \cdot [D_k - L + {F_j}^{\pi} - t_{i-1}] \cdot {y_j}^i \cdot {z_i}^k + \sum_{W_k \in \Gamma} h_k \cdot \sum_{i=1}^{\omega} {z_i}^k. \tag{3}$$

The sum over all jobs is the weighted flow time cost associated with B_i if it is processed in W_k whereas h_k is the booking cost. The finish time of job $J_j \in B_i$ in W_k in this case is the start of W_k; i.e. $D_k - L$, plus the additional time ${F_j}^{\pi} - t_{i-1}$ which is the the difference between the start t_{i-1} of B_i and the finish time ${F_j}^{\pi}$ of job J_j in the single-machine sequence π. We have the following

result:

Theorem 1 *Minimizing the total weighted flow time subject to immediate shipments is strongly NP-Hard.*

In light of Theorem 1, in what follows we propose three heuristics for the problem of minimizing weighted flow time plus booking costs subject to immediate shipments. All three heuristics start by arranging the jobs in Weighted-Shortest-Processing-Time (WSPT) order (simply putting jobs in decreasing $\frac{w_j}{p_j}$ order) and proceed in two steps: (i) Producing a sequence π (which in turn yields batching B_1,B_2,B_3,...,B_ω), and then (ii) finding an optimal collection of windows to allocate the batches. Our heuristics differ from one another with respect to step (i). For step (ii), they all use the same optimal dynamic program. We start by describing algorithms for producing batches B_1,B_2,B_3,...,B_ω.

The first batching procedure takes the WSPT order of jobs and batches them by applying (2) without further processing. The second batching procedure is called ω-*Partition* and solves a series of $\omega - 1$ knapsack problems as described next:

ω-Partition
Input: Processing times p_j and weights w_j for $J_j \in N = \{J_1, \ldots, J_n\}$.
Output: Batches B_1,B_2,B_3,...,B_ω.

[0] Let $k = \omega - 1$, B_1=B_2=B_3= ...=$B_\omega = \emptyset$ and $NU = N$

[1] Solve knapsack problem: $M(k) = \max \sum_{J_j \in NU} w_j \cdot x_j$ s.t. $\sum_{J_j \in NU} p_j \cdot x_j \leq k \cdot L$, $x_j \in \{0,1\}$
where x_j=1 if J_j is in the knapsack and 0 otherwise.

[2] Set $B_{k+1} = \{J_j : x_j = 0\}$, NU=NU - B_{k+1}
If k=0 then STOP, else set k=$k-1$ and go to [1].

Our third batching procedure is called ω-*Ratio-Partition* and it is identical to ω-*Partition* except for the objective function of the knapsack problem in line [1] which now becomes $\max \sum_{J_j \in NU} (w_j/p_j) \cdot x_j$. The number of batches ω is bounded by T and hence at most T knapsack problems will be solved for heuristic ω-*Partition*. Solving a knapsack problem requires $O(n \sum_{J_j \in N} p_j)$ time and therefore heuristic ω-*Partition* is pseudo-polynomial with complexity $O(nT \sum_{J_j \in N} p_j)$ as is ω-*Ratio-Partition*.

Given B_1,B_2,B_3,...,B_ω, the following dynamic programming formulation, referred to as BA, optimally assigns batches to manufacturing windows. Let $f_i(k)$ be the minimum total booking plus WIP costs for B_1,...,B_i when B_i is processed in window W_k, and B_1,...,B_{i-1} are processed prior to W_k.

Recursive relation:

$$f_i(k) = \min_{i-1 \le r < k} \{ f_{i-1}(r) + h_k + \sum_{J_j \in N} w_j \cdot [D_k - L + {F_j}^{\pi} - t_{i-1}] \cdot {y_j}^i \}, \quad for \quad i \le k \le T - \omega + i. \tag{4}$$

Boundary condition: $f_0(k) = 0$ for k=0 and ∞ otherwise. Optimal value: $f^* = \min_{\omega \le k \le T} \{ f_\omega(k) \}$.

The number of $f_i(k)$ states is bounded by ωT which is of order $O(T^2)$. Recurrence relation (4) is a minimization over at most T-i+1 values, which is of order $O(T)$. Therefore, the complexity of BA is $O(T^3)$.

The heuristic solutions produced by BA when the initial batching follows the WSPT rule, ω-*Partition* and ω-*Ratio-Partition* are called $WSPT$, $Knap$ and $RKnap$, respectively.

3.1 Lower Bounds

We develop lower bound (LB) schemes to help us evaluate the quality of our heuristics. To describe our first LB, we introduce a variation of our model where each job $J_j \in B_i$ is charged a cost of $w_j \cdot D_k$ (D_k = End of window W_k) if B_i is processed in W_k. We call this problem the *batch shipment problem* which is presented in detail in Aydinliyim (2007). The objective function for the *batch shipment problem* is identical to (3) except that $[D_k - L + {F_j}^{\pi} - t_{i-1}]$ is replaced with D_k such that:

$$Z^*_{batch} = \min_{\sigma} \sum_{W_k \in \Gamma} \sum_{i=1}^{\omega} \sum_{J_j \in N} w_j \, D_k \, {y_j}^i \, {z_i}^k + \sum_{W_k \in \Gamma} h_k \sum_{i=1}^{\omega} {z_i}^k. \tag{5}$$

In what follows we introduce a lower bounding scheme $MaxKnap$ for the batch shipment problem. $MaxKnap$ is identical to $Knap$ except that: (i) in line [2] of ω-*Partition* we do not perform NU=NU - B_{k+1} and, (ii) the dynamic program BA in (4) is replaced by

$$\tilde{f}_i(k) = \min_{i-1 \le r < k} \{ \tilde{f}_{i-1}(r) + h_k + \sum_{J_j \in N} w_j \, D_k \, {y_j}^i \}. \tag{6}$$

Let $Z_{MaxKnap}$ be the resulting objective function value when $MaxKnap$ is applied. Then, we have the following lemma:

Lemma 1 *$MaxKnap$ produces a lower bound for the batch shipment problem, i.e. $Z_{MaxKnap} \le Z^*_{batch}$.*

Let $w_i(k)$ be the sum of the weights of the jobs that belong to B_i in σ and G be the time lag between two consecutive manufacturing windows. Then, (3) can be rewritten as

$$\min_{\sigma} \sum_{J_j \in N} (\sum_{i=1}^{j} p_i) \cdot w_j + \sum_{i=1}^{\omega} \sum_{W_k \in \Gamma} w_i(k) \cdot ((k-1) \cdot G + (k-i) \cdot L) \cdot {z_i}^k + \sum_{W_k \in \mathcal{W}^\sigma} h_k. \tag{7}$$

The first term in (7) is the objective function value of the weighted flow time on a single machine. It is known that the WSPT order is optimal for $1 \,||\, \sum w_j \cdot C_j$ (see Smith (1956)). Let Z_{WSPT} be the minimum cost for this problem. The rest of (7) has the same form as the objective function of the *batch shipment problem* in (5) except that D_k is replaced by $(k-1) \cdot G + (k-i) \cdot L$. Let $Z_{MaxKnap}$ be the value when $MaxKnap$ is applied to the problem with the objective function $\sum_{i=1}^{\omega} \sum_{W_k \in \Gamma} w_i(k) \cdot [(k-1) \cdot G + (k-i) \cdot L] + \sum_{W_k \in \mathcal{W}^\sigma} h_k$. Note that both $\{D_k\}$ and $\{(k-1) \cdot G + (k-i) \cdot L\}$ are increasing sequences of k. Therefore, the same choice of weights $w_i(k) : W_k \in \Gamma$ minimize both $\sum_{i=1}^{\omega} \sum_{W_k \in \Gamma} w_i(k) \cdot D_k + \sum_{W_k \in \mathcal{W}^\sigma} h_k$ and $\sum_{i=1}^{\omega} \sum_{W_k \in \Gamma} w_i(k) \cdot [(k-1) \cdot G + (k-i) \cdot L] + \sum_{W_k \in \mathcal{W}^\sigma} h_k$. Also note that $MaxKnap$ provides a lower bound for the objective function of the *the batch shipment problem* (see Lemma 1). Then, we have the following theorem.

Theorem 2 *$Z_{WSPT} + Z_{MaxKnap}$ is a lower bound for the problem of minimizing the total weighted flow time plus booking costs subject to immediate shipments, i.e. $Z_{WSPT} + Z_{MaxKnap} \leq Z^*$.*

The proof of Theorem 2 follows from Lemma 1 and the above observation. Hence, it is omitted.

An alternative lower bound will now be developed by assuming that each delivery includes not only the batch jobs completed during a window, but also the completed portion of a preempted job. Let p_{jk} be the portion of p_j completed in W_k. Then to each job portion p_{jk}, we assign weight $f_{jk} = \frac{p_{jk}}{p_j} w_j$. Likewise, we assign p_{jk} fraction $\frac{p_{jk}}{L} h_k$ of the booking cost of W_k. To obtain a lower bound, we further assume that every batch is shipped at the beginning of its window W_k, i.e. at time $D_k - L$. For instance, if 2 units of a job with processing time $p_j = 5$ and weight w_j are processed in W_k and the remaining 3 units are processed in W_{k+1}, then the WIP cost for this job would be $\frac{2}{5} \cdot w_j \cdot (D_k - L) + \frac{3}{5} \cdot w_j \cdot (D_{k+1} - L)$ while the booking cost would be $\frac{2}{L} \cdot h_k + \frac{3}{L} \cdot h_{k+1}$.

Equivalently, $c_{jk} = (\frac{w_j}{p_j} \cdot (D_k - L) + \frac{h_k}{L})$ is the cost of processing one unit of J_j in W_k shipped at the beginning of W_k. Let u_k be 1 if W_k is utilized (either fully or partially); 0 otherwise. Then, the lower bounding scheme is equivalent to the linear program

$$\min \quad \sum_{J_j \in N} \sum_{W_k \in \Gamma} c_{jk} \cdot p_{jk} \tag{8}$$

$$s.t. \quad \sum_{W_k \in \Gamma} p_{jk} = p_j \quad , \forall\, J_j \in N \tag{9}$$

$$\sum_{J_j \in N} p_{jk} \leq L \cdot u_k \quad , \forall\, W_k \in \Gamma \tag{10}$$

$$\sum_{W_k \in \Gamma} u_k \leq \omega \tag{11}$$

$$u_k \in \{0\,,\,1\} \qquad ,\forall\, W_k \in \Gamma \tag{12}$$

$$p_{jk} \geq 0 \qquad ,\forall\, J_j \in N\,,\,\forall\, W_k \in \Gamma \tag{13}$$

where p_{jk} and u_k are variables. Constraints (9) assure that each job is fully processed over the windows booked. Constraints (10) ensure that no window is over-utilized. Constraint (11) ensures that precisely ω windows are used. For any collection of windows, it is optimal to process jobs in N according to the WSPT order of the job portions (see Smith (1956)). Once π is given, BA finds the optimal allocation of batches to windows. This lower bound scheme is polynomial because finding π takes $O(nlogn)$ time and running BA for the associated batches takes $O(T^3)$ time. This lower bounding scheme will be referred to as $FracWSPT$ and runs much faster than $MaxKnap$ which has complexity $O(nT\sum_{J_j \in N} p_j + T^3)$.

4 Coordination

After the manufacturers finish their bookings and create their own initial schedules ${\sigma_0}^m$, $m \in M$, they have the option to cooperate and form coalitions so as to decrease the total cost associated with the members of the coalition. During this process $3P$ acts as a controller who imposes rules on the formation of coalitions and creates a coordinating mechanism.

4.1 Game Definition

Consider the set of manufacturers, M, and the jobs $N = \{J_1, \ldots, J_n\} = \cup_{m \in M} N_m$ where N_m is the set of jobs that belong to manufacturer m. Similarly, N_S is the set of jobs that belong to members of coalition S. Let $\mathcal{W}_{\sigma_0}(S) \subseteq \Gamma$ denote the set of windows that are utilized in the initial schedule by the members of S (i.e., $\mathcal{W}_{\sigma_0}(S) = \cup_{m \in S} \mathcal{W}_{\sigma_0^m}$). Also let $\sigma^*(S)$ be the optimal schedule for coalition S and $\mathcal{W}_{\sigma^*}(S)$ be the corresponding set of windows utilized had manufacturers in S coordinated their jobs in N_S. Moreover, let $c_j(\sigma_0(S))$ and $c_j(\sigma^*(S))$ be the WIP costs associated with job $J_j \in N$ in $\sigma_0(S)$ and $\sigma^*(S)$, respectively. At this point we define the set of *admissible rearrangements* of jobs as follows.

Definition 1 *Let σ^S be a sequence of jobs in N_S. A sequence σ^S is said to be admissible with respect to $\sigma_0(S)$ if every $J_j \in N \backslash N_S$ has the same predecessors in both σ^S and $\sigma_0(S)$.*

We make the following assumptions about the cooperative game amongst the manufacturers:

1. Booking refunds are awarded to manufacturers if they agree to form the grand coalition M.

2. The optimal schedule for coalition S, namely $\sigma^*(S)$ must be admissible with respect to $\sigma_0(S)$.

3. The optimal collection of windows for coalition S, namely $\mathcal{W}_{\sigma^*}(S)$, is chosen from the set of windows initially booked by the members of coalition S, i.e. $\mathcal{W}_{\sigma^*}(S) \subseteq \mathcal{W}_{\sigma_0}(S)$.

Recall that $3P$ realizes benefits of coordination via booking refunds and resale of the manufacturing windows. $3P$ wants to maximize his possible savings by forcing the manufacturers to schedule their jobs according to the most compact schedule which utilizes the idle times across booked windows. Therefore, $3P$ maximizes his benefits when all manufacturers cooperate. Hence, Assumption 1 leverages $3P$'s refund policy and results to maximum savings for himself. Otherwise, manufacturers could coalesce in ways that do not optimize the utilization of manufacturing windows from the $3P$'s point of view. Assumption 2 is standard in most single machine sequencing games and implies that job rearrangements among members of coalition S do not effect the scheduling of jobs of members in $M \backslash S$. Assumption 3 means that members of a coalition S cannot utilize windows that were not booked by its members prior to coordination. This stipulation prevents two distinct coalitions from seeking access to the same window among those left un-booked in the initial schedule. Given the above definitions and assumptions, we define the cooperative game (M, υ) where $\upsilon : 2^{|M|} \rightarrow \mathbf{R}$, $\upsilon(\emptyset) = 0$ and

$$\upsilon(S) = \begin{cases} \sum\limits_{m \in S} \sum\limits_{J_j \in N_m} [c_j(\sigma_0(S)) - c_j(\sigma^*(S))] & \text{if } S \subset M \\ \sum\limits_{m \in S} \sum\limits_{J_j \in N_m} [c_j(\sigma_0(S)) - c_j(\sigma^*(S))] + \rho \cdot [\sum\limits_{W_k \in \mathcal{W}_{\sigma_0}(S)} h_k - \sum\limits_{W_k \in \mathcal{W}_{\sigma^*}(S)} h_k] & \text{if } S = M \end{cases} \tag{14}$$

where $0 \leq \rho \leq 1$ is the refund percentage. Evidently, $\upsilon(S)$: $S \subseteq M$ represents the total amount of savings that can be attained when all manufacturers in S coordinate to process jobs in N_S.

The optimal schedule for all manufacturers, say σ^* (or $\sigma^*(M)$), can only be utilized if all manufacturers agree to schedule their jobs as suggested by $3P$. For this to happen $\upsilon(M)$ must be allocated so that all manufacturers are better off following σ^* rather than using their previously reserved windows or forming smaller coalitions $S \subset M$. If we denote the amount of savings allocated to each manufacturer by x_m, $m \in M$, we seek an allocation vector, $\mathbf{x} = \{x_1, \ldots, x_{|M|}\}$ that satisfies the following *core* (in)equalities:

$$\sum_{m \in S} x_m \geq \upsilon(S) \ \forall\, S \subseteq M \ , \ \sum_{m \in M} x_m = \upsilon(M). \tag{15}$$

The first set of constraints are needed to ensure that every coalition S receives from $3P$ at least as much as they can save by themselves. The second equality guarantees that all possible savings (WIP savings and fraction ρ of the booking savings) generated by coordination are allocated to

the manufacturers. We assume that σ^* is acceptable by all manufacturers only if a core allocation exists.

Note that the cooperative game may become intractable when Assumptions 1 and 2 are relaxed. The following example demonstrates that the core can be empty in such a case. Consider 2-unit manufacturing windows each with booking price h and booking refund ρ. Without loss of generality, suppose that manufacturers A, B, and C have 1-unit workload each and initially book windows W_1, W_2, and W_3, respectively. For simplicity assume that the weights of all jobs are negligible. Then (without Assumptions 1 and 2), $\upsilon(\{A,B,C\}) = \upsilon(\{A,B\}) = \upsilon(\{B,C\}) = \upsilon(\{A,C\}) = \rho h$, and $\upsilon(\{A\}) = \upsilon(\{B\}) = \upsilon(\{C\}) = 0$. It is easy to see that there is no such allocation $\mathbf{x} = \{x_A, x_B, x_C\}$ that satisfies the core (in)equalities. When only Assumption 1 is relaxed, the value function values are $\upsilon(\{A,B,C\}) = \upsilon(\{A,B\}) = \upsilon(\{B,C\}) = \rho h$, and $\upsilon(\{A\}) = \upsilon(\{B\}) = \upsilon(\{C\}) = \upsilon(\{A,C\}) = 0$. Then, the only core allocation is $\mathbf{x} = \{x_A, x_B, x_C\} = \{0, \rho h, 0\}$. However, in such a situation manufacturers A and C are indifferent between cooperating or acting alone. Imposing Assumptions 1 and 2, the value functions become $\upsilon(\{A,B,C\}) = \rho h$ and $\upsilon(S) = 0$ for $S \neq M$. Using the allocation rule described in Section 4.2 enables $3P$ to allocate positive savings to each manufacturer (due to uniform allocation of the booking savings) so that coordination is strictly better.

4.2 Structural Results and Allocation Rules

In this section we develop properties of game (M, υ) that lead to the development of an allocation rule that satisfies (15). We start with the following definition.

Definition 2 *A game is said to be convex when*

$$\forall\, S, T \subseteq M\ ,\ \upsilon\{S \cup T\} + \upsilon\{S \cap T\} \geq \upsilon\{S\} + \upsilon\{T\}\ .$$

In (Shapley 1971) it is shown that *convex* games have non-empty core. The following example shows that (M, υ) is not necessarily convex.

Example 1: Consider a 4-player game in which each player has one job to process, $M = \{1,2,3,4\}$, N_1=$\{J_1\}$, N_2=$\{J_2\}$, N_3=$\{J_3\}$, N_4=$\{J_4\}$ p_1=2, p_2=1, p_3=3, p_4=1, w_1=w_2=w_3=w_4=1, $|\Gamma|$=4, L=3, ρ=1, h_1=h_2=h_3=h_4=0 and each window starts 3 time units after the completion of the previous one ($G = 3$). The initial schedule costs $1 \cdot 2 + 1 \cdot 7 + 1 \cdot 15 + 1 \cdot 19 = 43$. The optimal schedule for M costs $1 \cdot 1 + 1 \cdot 2 + 1 \cdot 7 + 1 \cdot 13 = 23$. Therefore, $\upsilon(M)$=$\upsilon(\{1,2,3,4\})$= 20. Similar calculations show that $\upsilon(\{1,2,3\})$=11, $\upsilon(\{2,3,4\})$=12, and $\upsilon(\{2,3\})$=2. Now consider, S=$\{1,2,3\}$, T=$\{2,3,4\}$, $S \cup T$=$\{1,2,3,4\}$ and $S \cap T$=$\{2,3\}$. Then, $\upsilon(\{1,2,3\}) + \upsilon(\{2,3,4\}) = 23 > 22 = \upsilon(\{1,2,3,4\}) + \upsilon(\{2,3\})$ and hence the game is not convex. □

However, (M, υ) possesses the following property.

Definition 3 *Game (M,υ) is said to be superadditive when*

$$\forall\, S, T \subseteq M \text{ with } S \cap T = \emptyset, \upsilon\{S \cup T\} \geq \upsilon\{S\} + \upsilon\{T\}.$$

This means that two disjoint coalitions can do no worse by forming a larger coalition which is just the union of the two. Consider coalition S, the booking savings $\upsilon_h(S)$ and the WIP savings $\upsilon_c(S)$. Then,

$$\upsilon_c(S) = \sum_{m \in S} \sum_{J_j \in N_m} [c_j(\sigma_0(S)) - c_j(\sigma^*(S))], \quad \textit{and} \tag{16}$$

$$\upsilon_h(S) \quad = \begin{cases} 0 & \text{, if } S \subset M \\ \sum\limits_{W_k \in \mathcal{W}_{\sigma_0}(S)} h_k - \sum\limits_{W_k \in \mathcal{W}_{\sigma^*}(S)} h_k & \text{, if } S = M. \end{cases} \tag{17}$$

Proposition 1 *Games (M,υ), (M,υ_c) and (M,υ_h) are superadditive.*

Proof of Proposition 1: Consider coalitions $S, T \subseteq M$ with $S \cap T = \emptyset$ and let $\upsilon_c(S) + \upsilon_c(T)$ be the cost savings incurred when S uses $\mathcal{W}_{\sigma^*(S)}$ and T uses $\mathcal{W}_{\sigma^*(T)}$. Then, coalition $S \cup T$ can use windows in $\mathcal{W}_{\sigma^*(S)} \cup \mathcal{W}_{\sigma^*(T)}$ and reshuffle the jobs that belong to members in $S \cup T$ to obtain an improved schedule with savings $\upsilon_c(S \cup T) \geq \upsilon_c(S) + \upsilon_c(T)$. Hence, (M, υ_c) is superadditive. Naturally, (M, υ_h) is also superadditive because only $\upsilon_h(M)$ is non-zero. By definition, $\upsilon(S) = \upsilon_c(S)$ for $S \subset M$ and $\upsilon(M) = \upsilon_c(M) + \rho \cdot \upsilon_h(M)$ and hence the superadditivity of (M, υ_c) implies that (M, υ) is superadditive. This completes the proof of the Proposition. □

A clarification for Assumption 3 is in order. Recall that in this game setting, members of a coalition cannot utilize windows that were not booked by its members prior to coordination. This prevents two distinct coalitions from seeking access to the same un-booked window from the initial schedule σ_0. It is easy to construct examples where this stipulation is removed and Assumption 1 is relaxed but the booking game is not superadditive.

WIP savings for a coalition $S \subset M$ can be attained by two means: (a) utilization of idle times in $\mathcal{W}_{\sigma_0(S)}$, and (b) job rearrangements. Let, $\upsilon_{id}(S)$ and $\upsilon_{pw}(S)$ be the savings due to (a) and (b) respectively. Then,

$$\upsilon_c(S) = \upsilon_{id}(S) + \upsilon_{pw}(S) \ , \forall\ S \subseteq M. \tag{18}$$

Let $S \subseteq M$ be an arbitrary coalition. For ease of presentation, rather than working with coalitions of manufacturers, we would like to work with the corresponding window coalition $\mathcal{W}_{\sigma_0}(S)$. Let [a,b] denote the set of windows $\{W_a, W_{a+1}, W_{a+2}, \ldots, W_b\}$. Then, one can express $\mathcal{W}_{\sigma_0}(S)$ as

$$\mathcal{W}_{\sigma_0}(S) = [a_1, b_1] \cup [a_2, b_2] \cup \ldots \cup [a_r, b_r].$$

where $1 \leq a_1 < b_1 < a_2 < \ldots < a_r < b_r \leq T$. We refer to $[a_1, b_1], [a_2, b_2]$ etc. as *maximally connected* components of the coalition $\mathcal{W}_{\sigma_0}(S)$ of windows. Next, we rewrite the value function in terms of maximally connected components and use them to define a core allocation.

Savings produced by the windows in $[a, b]$ consist of savings $\omega_{id}([a, b])$ due to better utilization of idle times and savings $\omega_{pw}([a, b])$ due to job rearrangements in (18). Therefore,

$$\omega_c([a, b]) = \omega_{id}([a, b]) + \omega_{pw}([a, b]). \tag{19}$$

Let us define the following:

I_k = the idle time available in W_k in schedule σ_0; $I_k = 0$ if $W_k \notin \mathcal{W}_{\sigma_0}$,

s_k = the sum of the weights of jobs processed in W_k in schedule σ_0; $s_k = 0$ if $W_k \notin \mathcal{W}_{\sigma_0}$,

$N_{[a,b]}$ = the set of jobs that complete in windows $W_k \in [a, b]$ in the initial schedule,

$\sigma_0([a,b])$ = initial schedule for jobs initially processed in windows [a,b],

$\sigma^*([a,b])$ = optimal schedule for jobs initially processed in windows [a,b].

Then we can express $\omega_c([a, b])$ and $\omega_{id}([a, b])$ as follows:

$$\begin{array}{c} \omega_c([a, b]) = \sum_{J_j \in N_{[a,b]}} [c_j(\sigma_0([a, b])) - c_j(\sigma^*([a, b]))] \\ \omega_{id}([a, b]) = \sum_{W_k \in [a,b]} I_k \cdot (\sum_{W_t \in [k,b]} s_t). \end{array} \tag{20}$$

Then, one can express the savings of a coalition of manufacturers in terms of savings of a coalition of windows as follows:

$$\upsilon_{id}(S) = \sum_{k=1}^{r} \omega_{id}([a_k, b_k]), \quad \upsilon_{pw}(S) = \sum_{k=1}^{r} \omega_{pw}([a_k, b_k]), \quad \upsilon_c(S) = \sum_{k=1}^{r} (\omega_{id}([a_k, b_k]) + \omega_{pw}([a_k, b_k])). \tag{21}$$

Now we can develop allocation rules that make every manufacturer better off as part of the grand coalition than by forming smaller coalitions. Let ${x_m}^{wip}$ be the WIP savings allocation offered by $3P$ to manufacturer m. For $0 \leq \lambda \leq 1$ define

$$\begin{aligned} {x_m}^{id} &= \sum_{W_k \in \mathcal{W}_{\sigma_0 m}} [\frac{1}{2} \cdot I_k \sum_{W_t \in [k+1,T] \cap \mathcal{W}_{\sigma_0}} s_t + \frac{1}{2} \cdot s_k \sum_{W_t \in [1,k-1] \cap \mathcal{W}_{\sigma_0}} I_t] \\ {x_m}^{pw} &= \sum_{W_k \in \mathcal{W}_{\sigma_0 m}} [\lambda \cdot (\omega_{pw}([1, k]) - \omega_{pw}([1, k-1])) + (1 - \lambda) \cdot (\omega_{pw}([k, T]) - \omega_{pw}([k+1, T]))] \\ {x_m}^{wip} &= {x_m}^{id} + {x_m}^{pw} \end{aligned} \tag{22}$$

The intuition behind this allocation is as follows. For idle time savings, window W_k contributes I_k units of idle time to the completion time of jobs completing in successive windows. The total weight of these jobs is $\sum_{W_t \in [k+1,T] \cap \mathcal{W}_{\sigma_0}} s_t$ and therefore the resulting contribution of I_k to the WIP savings is $I_k \cdot \sum_{W_t \in [k+1,T] \cap \mathcal{W}_{\sigma_0}} s_t$. Half of those savings are allocated to the manufacturer who owns W_k leaving the rest for the owners of the windows that follow. By symmetry, all jobs in W_k (whose total weight is s_k) benefit by starting $\sum_{W_t \in [1,k-1] \cap \mathcal{W}_{\sigma_0}} I_t$ time units earlier. The owner of W_k receives half of those savings as well. Note that the availability of I_k units of idle time may result to completing a job at an earlier window in which case the completion time decreases by at least $I_k + G$. The contribution of G to the WIP savings is accounted for in ${x_m}^{pw}$. The rationale behind ${x_m}^{pw}$ is that the owner of window W_k receives portion λ of the marginal savings achieved by W_k when it joins the coalition of all preceding windows plus portion $1 - \lambda$ of the marginal savings achieved when it joins the coalition of all following windows.

The idle time savings are split equally between the owner of the windows and the other manufacturers. The rationale for this split is that value creation by accommodating future jobs by using the idle time of W_k is equally important to value creation by utilizing the idle time of earlier windows to process the workload in W_k. On the other hand the savings due to job rearrangements are split in proportion to λ and $1 - \lambda$. Here, we allow differentiating between the marginal benefits resulting by adding W_k to the tail of the window coalition $[1, k-1]$ rather than to the head of the window coalition $[k+1, T]$. In other words, adding W_k to $[k+1, T]$ provides different savings opportunities than adding W_k to $[1, k-1]$. The former action may provide flexibility to all the jobs in $[k+1, T]$ whereas the latter provides flexibility only to jobs in W_k. We can now prove the following result.

Theorem 3 *Expressions in* (22) *provide a core allocation for the game* (M, υ_c).

The third party $3P$ allocates additional savings to the manufacturers that are made possible via booking refunds. Recall that ${x_m}^{wip} : m \in M$ is already a core allocation, so any addition would also be a core allocation for games (M, υ) as booking savings are offered by the $3P$ only to the grand coalition. One such allocation is attained if additional savings are distributed uniformly amongst the manufacturers. Therefore,

Theorem 4 *Allocation*

$$x_m \quad = {x_m}^{wip} + \rho \cdot \tfrac{1}{|M|} \cdot (\upsilon(M) - \upsilon_c(M)), \quad m \in M \tag{23}$$

is in the core of game (M, υ).

5 Computational Results and Managerial Insights

In this section we perform a computational experiment to test the heuristics proposed in the preceding sections. Then we use these heuristics to assess the value of coordination in our model.

5.1 Performance of Algorithms

In this experiment we generate problem instances by varying the values of parameters L, G, h_k for $k = 1, \ldots, T$, p_j and w_j for $J_j \in N$. We consider the values $L \in \{8, 16\}$ for the length of a manufacturing window, corresponding to one or two 8-hour shifts of production capacity. The gap G between windows is 24-L and it represents the off-production hours. We consider a production planning horizon of 100 days. We assume that 20 days during the planning horizon are unavailable due to earlier reservations, non-working days, etc. The booking cost of a window is $h_k = h \cdot (T - k)$ where h is drawn uniformly from [1,150]. This implies that booking prices decrease on average as k increases placing higher premium on early windows. The processing times p_j are drawn uniformly from [1,4] or [1,12]. This enables us to compare four different scenarios for the mean number $\frac{L}{\bar{p_j}}$ of jobs per window varying from 1.23 to 6.40, where $\bar{p_j}$ is the mean length of a job. Equivalently, these scenarios capture applications with medium to large manufacturing jobs. Large jobs result in a smaller ratio $\frac{L}{\bar{p_j}}$ which enables us to examine scenarios where preemptions occur more frequently. Weight w_j for $J_j \in N$ is drawn uniformly from [1,10] or [1,20]. The problem size represented by the number of jobs $|N|$ takes on the values 20, 40, 60, 80 or 100. Therefore, the average number of windows needed to process all jobs of a manufacturer varies from 3 (when $p_j \in [1, 4]$, $L = 16$ and $n = 20$) to 75 (when $p_j \in [1, 12]$, $L = 8$ and $n = 100$) which allows us to test the performance of heuristics under diverse workload scenarios.

The above selection of parameters results to a total of 40 combinations. We run 10 instances for each combination of parameters using heuristics $Knap$, $RKnap$, $WSPT$ and the lower bounds $MaxKnap$, $FracWSPT$. For each instance we denote by LB, LB^* the smaller and the larger of these two lower bounds respectively and by Z_H the cost associated with heuristic H in {$Knap$, $RKnap$, $WSPT$}. We report averages over the 10 instances of the following statistics:

$\frac{Z_H - LB^*}{LB^*}$ 100%: the relative percentage deviation of Z_H from LB^*,

$\frac{LB^* - LB}{LB^*}$ 100%: the relative percentage deviation of LB from the tighter lower bound LB^*.

Table 1 presents our findings. We found that the performance of algorithms did not depend on the w_j values. The difference between the relative percentage deviation values, $\frac{Z_H - LB^*}{LB^*}$, for $w_j \in [1, 10]$ and $w_j \in [1, 20]$ are not statistically significant for our heuristics $Knap$, $RKnap$, and $WSPT$ with p-values 0.073, 0.118, and 0.697 respectively. For this reason, we present aggregate statistics over the 2 possible choices of w_j values. Based on the above experiments we make the

Table 1: Performance of Heuristics and Lower Bounds

Parameters			$\frac{Z_H - LB^*}{LB^*}$ 100%			$\frac{LB^* - LB}{LB^*}$ 100%
p_j	L	$\|N\|$	*Knap*	*RKnap*	*WSPT*	*FracWSPT*
[1,4]	8	20	0.12	1.39	1.42	2.43
		40	0.67	2.08	1.92	1.20
		60	0.18	1.80	0.82	1.47
		80	0.12	1.57	0.67	1.26
		100	0.18	1.62	0.50	1.11
	16	20	0.04	1.45	1.41	5.17
		40	0.04	1.38	1.13	4.43
		60	0.06	1.20	0.48	4.25
		80	0.05	1.43	0.60	3.78
		100	0.05	1.84	0.41	3.38
[1,12]	8	20	2.98	1.58	1.83	2.88
		40	3.07	1.18	1.53	1.45
		60	1.72	0.84	1.01	1.05
		80	2.54	0.67	0.83	0.86
		100	2.06	0.49	0.61	0.78
	16	20	1.00	1.37	1.61	4.24
		40	1.03	1.12	1.19	3.15
		60	1.28	0.70	0.79	2.45
		80	1.21	0.74	0.62	2.09
		100	0.91	0.63	0.50	1.85
	Average		0.97	1.25	0.99	2.46

following observations:

- For $p_j \in$ [1,4], *Knap* dominates all other heuristics.

When jobs in N are small, *Knap* outperforms the other heuristics. This is because at the end of every manufacturing window *Knap* gives priority to unscheduled jobs that fit within the window and have large weight (e.g. a job with p_j=3 and w_j=10). On the other hand *RKnap* gives priority to multiple unscheduled jobs that are smaller but have large $\frac{w_j}{p_j}$ ratio (e.g. 3 jobs with p_j=1 and w_j=3). Evidently, this gives a slight advantage to *Knap* when the mean processing time of a job is small and hence the number of jobs per window is large. This is illustrated in Table 1 with all combinations where $p_j \in$ [1,4].

- *RKnap* dominates for L=8 and $p_j \in$ [1,12].

Heuristic *Knap* loses the above-mentioned advantage as the number of jobs per manufacturing window decreases. This is precisely the scenario with L=8 and $p_j \in$ [1,12] where *RKnap* outperforms *Knap* (see Table 1) as *RKnap* has the advantage when the average number of jobs per window is small. Evidently, *RKnap* makes a better distribution of small jobs across the schedule and avoids the delays caused by large jobs early in the schedule. Moreover, when L=16 and $p_j \in$ [1,12], no heuristic dominates the other.

The difference between the relative percentage deviation of LB from the tighter lower bound , $\frac{LB^*-LB}{LB^*}$, for $w_j \in [1,10]$ and $w_j \in [1,20]$ is significantly different with p-value 0.001, but for sake of uniformity in Table 1 we reported aggregated results as the following domination result is more important than the magnitude of the difference.

- Lower bound $MaxKnap$ dominates $FracWSPT$.

We should note that, despite its poorer performance, $FracWSPT$ has the advantage of being a polynomial lower bound scheme.

5.2 The Value of Coordination

The focus of our second experiment is (i) to quantify the benefits of coordination and (ii) to assess the value of our allocation scheme to the manufacturers from an individual perspective. This is done in subsections 5.2.1 and 5.2.2 respectively. Here we let (L, G)=(16,32) where each time unit is considered to be 30 minutes. The planning horizon is again T=100 days and we consider two booking prices: h_r and $h_p = 1.3h_r$. Price h_r corresponds to booking costs during regular customer demand for the manufacturer while h_p is used for peak demand periods. We select the value h_r so that the scheduler is indifferent between choosing an expensive window W_k and a cheap window which starts 4 days later, i.e. W_{k+4}. In other words, we choose h_r so that

$$h_k + \bar{w_j} \cdot \frac{L}{\bar{p_j}} \cdot [(k-1) \cdot G + k \cdot L] = h_{k+4} + \bar{w_j} \cdot \frac{L}{\bar{p_j}} \cdot [(k+3) \cdot G + (k+4) \cdot L]$$

where $\bar{w_j}$, $\bar{p_j}$ are the mean w_j- and p_j-values. Simple algebra yields

$$h_r = \frac{4}{0.3} \bar{w_j} \cdot \frac{L}{\bar{p_j}} \cdot (G + L).$$

Then, we randomly draw 15% or 30% of the windows and designate them as peak windows. Let $\mathcal{W}_P$ denote the set of peak windows. Then, $|\mathcal{W}_P|/T \in \{0.15, 0.30\}$. Processing times p_j are drawn uniformly from [1,4] for small jobs and from [5,8] for longer jobs. Therefore, the mean $\bar{p_j}$ is 75 minutes for a small job and 195 minutes for a large job. We experimented with the pairs $(4, 20)$ and $(8, 10)$ for $(|M|, |N_m|)$ so as to assess the effect of the number of windows per manufacturer to system profits and the effect of the size of the grand coalition. This set of parameters results to 8 combinations. For each combination 10 randomly generated problems are generated.

5.2.1 Overall Coordination Benefits

The following statistics are reported in Table 2:

Table 2: Cost Savings due to Coordination

Parameters			$\frac{\lvert\mathcal{W}_P\rvert}{\lvert\Gamma\rvert}=15\%$			$\frac{\lvert\mathcal{W}_P\rvert}{\lvert\Gamma\rvert}=30\%$		
$\lvert M\rvert$	$\lvert N_m\rvert$	p_j	ΔC	ΔB	A^+	ΔC	ΔB	A^+
4	20	[1,4]	30.73	10.40	72.50	28.16	10.65	65.00
		[5,8]	24.54	4.68	67.50	23.27	4.39	67.50
8	10	[1,4]	39.82	21.19	68.75	39.39	20.70	68.75
		[5,8]	30.84	10.59	63.75	36.89	10.10	63.75
Averages			32.73	11.72	68.13	31.93	11.46	66.25

$\Delta C = \frac{\sum_{m\in M}\sum_{J_j\in N_m}[c_j(\sigma_0)-c_j(\sigma_H)]+\sum_{k\in\mathcal{W}_{\sigma_0}}h_k-\sum_{k\in\mathcal{W}_{\sigma_H}}h_k}{\sum_{m\in M}\sum_{J_j\in N_m}c_j(\sigma_0)+\sum_{k\in\mathcal{W}_{\sigma_0}}h_k}100\%$; percentage savings due to coordination,

$\Delta B = \frac{\sum_{k\in\mathcal{W}_{\sigma_0}}h_k-\sum_{k\in\mathcal{W}_{\sigma_H}}h_k}{\sum_{k\in\mathcal{W}_{\sigma_0}}h_k}100\%$; percentage booking savings due to coordination, and

$A^+ = \frac{|\{m\in M:\sum_{J_j\in N_m}c_j(\sigma_H)<\sum_{J_j\in N_m}c_j({\sigma_0}^m)\}|}{|M|}100\%$; percentage of manufacturers whose WIP costs decrease after coordination,

where σ_H is the best among the 3 heuristic schedules produced. In Table 2, we aggregate savings statistics over the combinations (4,20), (8,10) for ($|M|$, $|N_m|$). We make the following observations:

- *Savings ΔC increase with $|M|$ (at a decreasing rate) and with $\frac{L}{\bar{p}_j}$.*

As the number $|M|$ of manufacturers increases the savings due to booking and due to job rearrangements increase because more idle time is freed up and more rearrangement opportunities are created. The overall average savings aggregated over $|M|$=4 and 8 are 27.93% and 36.74% respectively. Evidently, ΔC increases at a decreasing rate because when $|M|$ doubles ΔC does not. Even though it is not apparent from Table 2 due to aggregation, for L=16 and $p_j \in$ [1,4] or $p_j \in$ [5,8] we have $\frac{L}{\bar{p}_j}$ taking on the values 6.40 and 2.46 respectively with corresponding overall average ΔC values 34.53% and 30.14%. Hence, ΔC increases with $\frac{L}{\bar{p}_j}$.

- *Booking savings ΔB increase with $|M|$, decrease with p_j, but do not depend on $\frac{|\mathcal{W}_\mathcal{P}|}{T}$.*

As the number $|M|$ of manufacturers increases booking savings due to coordination increase because of better utilization of idle times and the release of unused windows. For $|M|$=4,8 the overall average booking savings are 7.53% and 15.65% respectively. On the other hand, for $p_j \in$ [5,8] the average ΔB value is 7.44%. This is about half the corresponding average for $p_j \in$ [1,4] which is 15.74. The above hold true for both $\frac{|\mathcal{W}_\mathcal{P}|}{T}$ values.

- *On average, after coordination 67.19% of the manufacturers have incurred lower WIP costs after coordination.*

As indicated by A^+, under all scenarios approximately two-thirds of the manufacturers improve over their initial schedules in terms of their WIP costs before the savings are allocated. These manufacturers are primarily late comers in the FCFS order because after coordination they use windows booked by early comers that were not available when they arrived to the system. Generally, the manufacturers who initially booked the later windows benefit from coordination as they gain access to earlier windows. Similarly, manufacturers who had the chance to book earlier windows might be delayed in the coordinated schedule, but the deterioration in their service level is compensated by the allocation of the savings as is demonstrated in the following subsection.

5.2.2 Allocation Results

The second part of our analysis is meant to provide insights on the value of the allocation scheme to individual manufacturers. Individual manufacturers' benefits due to coordination are summarized in Table 3 using the following statistics:

$\frac{\Delta WIP}{TC} = \frac{\sum_{J_j \in N_m} [c_j(\sigma_0{}^m) - c_j(\sigma_H)]}{\sum_{J_j \in N_m} c_j(\sigma_0) + \sum_{W_k \in \mathcal{W}_{\sigma'}} h_k} 100\%$; WIP savings as a percentage of total savings for manufacturer m after coordination,

${x_m}^{pw}$, ${x_m}^{id}$, ${x_m}^{wip}$, and x_m: defined in Section 4. We report each of these statistics as a percentage of the total savings in each category in the rows of Table 3 indicated by x^{pw}, x^{id}, x^{wip} and x respectively (e.g. x^{pw} row reports ${x_m}^{pw} / \sum_{m \in M} {x_m}^{pw} \cdot 100$ values). We make the following observations:

- *x^{wip} and x decrease with the manufacturer index m.*

About 32.81% of the manufacturers have worse WIP performance in the coordinated schedule. Hence it is fair that the portion of WIP savings allocated to them are higher than those for late comers in the FCFS order. For these manufacturers observe that the WIP savings as a percentage of the initial cost $\frac{\Delta WIP}{TC}$ increase with m (see Table 3). Because late comers improve dramatically with respect to WIP performance (eg. more than 50% for m=$|M|$), the portion of WIP savings allocated to them are small. (e.g., about 5-10% for $|M|$=4,8 respectively) and most of the savings are allocated to early comers. On the other hand x^{wip} and x decrease as $|M|$ increases because the total savings are distributed amongst more manufacturers.

- *The idle time savings x^{id}, decrease linearly with $|M|$. For small $|M|$, the x^{id} values follow a convex pattern which disappears when $|M|$ increases.*

The overall average x^{id} values for $|M|$=4,8 are 25.00% and 12.50% respectively. For $|M|$=4 the idle time savings seem to follow a convex pattern while for $|M|$=8 no pattern is evident (see Table

Table 3: Allocation Results

	$p_j = [1,4]$				$p_j = [5,8]$			
$\|M\|$=4	1	2	3	4	1	2	3	4
x^{pw}	40.95	26.10	21.79	11.16	41.90	27.60	20.15	10.36
x^{id}	30.06	16.50	25.49	27.95	33.68	21.63	21.16	23.53
x^{wip}	39.93	25.19	22.14	12.74	41.42	27.25	20.20	11.12
x	37.78	25.17	22.55	14.50	40.79	27.17	20.39	11.66
$\frac{\Delta WIP}{TC}$	-53.83	5.86	33.41	53.48	-124.33	5.49	33.67	53.25
	$p_j = [1,4]$							
$\|M\|$=8	1	2	3	4	5	6	7	8
x^{pw}	19.83	16.00	16.11	14.04	11.64	9.53	7.73	5.14
x^{id}	13.95	9.82	11.21	13.15	13.07	13.52	12.75	12.53
x^{wip}	19.07	15.20	15.48	13.92	11.82	10.04	8.38	6.09
x	17.72	14.65	14.87	13.63	11.96	10.55	9.22	7.41
$\frac{\Delta WIP}{TC}$	-57.91	-15.47	-5.12	28.26	31.87	49.49	49.98	55.96
	$p_j = [5,8]$							
$\|M\|$=8	1	2	3	4	5	6	7	8
x^{pw}	19.34	19.05	15.81	12.52	11.85	9.39	7.00	5.03
x^{id}	13.33	14.89	13.20	10.41	10.83	12.95	11.54	12.85
x^{wip}	18.84	18.70	15.59	12.34	11.77	9.69	7.39	5.68
x	18.40	18.28	15.38	12.35	11.82	9.89	7.74	6.15
$\frac{\Delta WIP}{TC}$	-164.51	-47.11	-6.02	20.44	23.68	48.72	51.95	56.63

3). The latter is because every manufacturer has several preceding and/or following windows that have idle time in the original schedule and hence he is allocated about the same amount of savings irrespective of his original position. In this case the savings due to idle times range between 9.82% and 14.89% for $|M|$=8 whereas for $|M|$=4 they range between 16.50% and 33.68%.

6 Conclusion and Future Research

In this study, we investigated a production chain coordination model where a number of manufacturers outsource similar operations to a single third party. We presented three heuristics and two lower bounds for the related optimization problem. Given the initial schedule, savings can be generated by coordinating production over the windows booked by all manufacturers in M. We presented allocation rules to share these savings so that every manufacturer agrees to follow the coordinated schedule. The highlight of our findings is that

"*the costs of the production chain can be reduced by an average of 32% percent if one-third of the members let* 3P *cover their increased WIP cost in exchange for 40-55% stake in the total savings.*"

Our future research will focus on different production settings (e.g. significant set-up times) and/or overtime availability. Also, models that consider the third-party as a player who has a share

of coordination savings in addition to booking refunds and re-booking earnings can be considered. Instead of the FCFS order for manufacturer bookings one can consider various possible booking disciplines. Examples include simultaneous booking of manufacturing windows as a result of an auction, sequential bookings by a series of auctions, or allowing for competition among manufacturers under an incentive rule suggested by the third-party. Furthermore, it is interesting to study the cooperative and competitive strategies of manufacturers under different information sharing protocols other than complete information.

Another possible direction is the study of outsourcing models where each manufacturer allocates his workload among his dedicated production resource and the flexible third-party resource which is common to other customers. In these models one can consider different contracts that determine the rules of engagement between the manufacturers and the third-party. Such contracts have been studied extensively in the supply chain coordination literature (see Cachon (2003)) which related to the coordination of inventories. However, no research has considered the design of contracts which model coordination of operations at the shop floor level.

Appendix

Proof of Theorem 1: The proof is based on a reduction from $3-partition$ which can be stated as follows. Given $3n$ integers in $A = \{a_1, \ldots, a_{3n}\}$ such that $B/4 < a_j < B/2$ and $\sum_{j=1}^{3n} a_j = nB$, is there a partition of A to n disjoint sets each having three elements that sum up to B?

Given an instance of $3-partition$, construct the following instance of the immediate shipment problem:

$$w_j = p_j = a_j \text{ , for } j = \{1,\ldots,3n\}$$
$$L = B \text{ , } h_1 = \ldots = h_n = 0$$

Let G be the time lag between two consecutive manufacturing windows. We will show that there exists a solution for the immediate shipment problem with cost z_{IS} if and only if there exists a solution to $3-partition$. First, suppose a solution for the immediate shipment problem exists. In this instance the booking costs are zero. Let s_k be the sum of the weights of the jobs in W_k. Then the problem of minimizing WIP costs with immediate shipment can be defined as follows:

$$
\begin{aligned}
\min \quad & z_{IS} = \sum_{j=1}^{3n} (\sum_{i=1}^{j} w_i) \cdot w_j + \sum_{k=1}^{n} s_k \cdot (k-1) \cdot G \\
s.t. \quad & s_1 \leq L \\
& s_1 + s_2 \leq 2L \\
& s_1 + s_2 + s_3 \leq 3L \\
& \quad \vdots \\
& s_1 + s_2 + s_3 + \ldots + s_n = nL
\end{aligned}
$$

The first double summation in the objective function is constant irrespective of the processing order of the jobs, because $\frac{w_j}{p_j} = 1$ by construction. The following is true for the second summation:

$$\sum_{k=1}^{n} s_k \cdot (k-1) \cdot G = \sum_{k=1}^{n} s_k \cdot k \cdot G - \sum_{k=1}^{n} s_k \cdot G = G \cdot (\sum_{k=1}^{n} s_k \cdot k) - n \cdot L \cdot G$$

Therefore, $z^* = \min z$ is attained when $\sum_{k=1}^{n} s_k \cdot k$ is minimized subject to the above constraints. Clearly, $z^* = \sum_{j=1}^{3n} (\sum_{i=1}^{j} w_i) \cdot w_j + \frac{(n-1)nLG}{2}$ if and only if ${s_1}^* = {s_2}^* = {s_3}^* = \ldots = {s_n}^* = L$. This means that the total weight and the total processing time requirements of the jobs that complete in each window is L. Since the constructed instance imposes the restriction $L/4 < p_j = w_j < L/2$ on the jobs, the optimal solution can be attained only if there are exactly three jobs finished in each window. Thus a solution for *the immediate shipment problem* induces a $3-partition$.

On the other hand, given a solution for $3-partition$, each part will have three elements with total processing time equal to L. Then, the 3 jobs associated with each part have total processing L, i.e. $s_1 = s_2 = s_3 = \ldots = s_n = L$. The total delivery cost of the resulting schedule is $z^* = \sum_{j=1}^{3n} (\sum_{i=1}^{j} w_i) \cdot w_j + \frac{(n-1)nLG}{2}$. Thus, a solution for $3-partition$ implies a solution for *the immediate shipment problem.* □

Proof of Lemma 1: Define $BW_i = M(i) - M(i-1)$ from (6) for i=1,2,3,...,ω. Suppose the dynamic program in (6) utilizes ω windows ending at D_1,D_2,D_3, ...,D_ω to process batch weights BW_1,BW_2,BW_3, ...,BW_ω. Likewise, suppose that an optimal schedule for *the batch shipment problem* utilizes windows with completion times ${D_1}^*$,${D_2}^*$,${D_3}^*$, ...,${D_\omega}^*$ to process batch weights ${W_1}^*$,${W_2}^*$,${W_3}^*$, ...,${W_\omega}^*$ respectively. We want to show that the total cost attained by $MaxKnap$, $Z_{MaxKnap}$, is less than the total cost associated with the optimal schedule, Z^*_{batch} i.e.

$$Z_{MaxKnap} = BW_1 \cdot D_1 + \ldots + BW_\omega \cdot D_\omega + h_1 + \ldots + h_\omega \leq {W_1}^* \cdot {D_1}^* + \ldots + {W_\omega}^* \cdot {D_\omega}^* + {h_1}^* + \ldots + {h_\omega}^*$$

where h_i and ${h_i}^*$ denote the booking cost of the i^{th} window booked by the heuristic and the optimal

schedules, respectively. Batch weights BW_1,BW_2,BW_3, ...,BW_ω are the maximum weight that can be processed within $L, 2L, 3L, \ldots, \omega L$ hours, and hence the following inequalities hold:

$$\begin{aligned}
BW_1 \geq {W_1}^* \Rightarrow \quad & \textstyle\sum_{J_j \in N} w_j - BW_1 \leq \sum_{J_j \in N} w_j - {W_1}^* \\
BW_1 + BW_2 \geq {W_1}^* + {W_2}^* \Rightarrow \quad & \textstyle\sum_{J_j \in N} w_j - BW_1 - BW_2 \\
& \leq \textstyle\sum_{J_j \in N} w_j - BW_1 - BW_2 \\
& \vdots \\
BW_1 + \ldots + BW_\omega \geq {W_1}^* + \ldots + {W_\omega}^* \Rightarrow \quad & \textstyle\sum_{J_j \in N} w_j - {W_1}^* - \ldots - {W_\omega}^* \\
& \leq \textstyle\sum_{J_j \in N} w_j - {W_1}^* - \ldots - {W_\omega}^*
\end{aligned} \tag{24}$$

Furthermore, the dynamic program in (6) finds the best windows to process batch weights BW_1, ...,BW_ω. Therefore $MaxKnap$ finds the minimum total cost of processing BW_1, ...,BW_ω, and hence the following inequality holds:

$$BW_1 \cdot D_1 + \ldots + BW_\omega \cdot D_\omega + h_1 + \ldots + h_\omega \leq BW_1 \cdot {D_1}^* + \ldots + BW_\omega \cdot {D_\omega}^* + {h_1}^* + \ldots + {h_\omega}^*$$

Otherwise, the solution of (6) could be improved by selecting the same windows as those used in the optimal schedule. The right hand side of the above inequality can be re-written as:

$$\begin{aligned}
& {BW_1}^* \cdot {D_1}^* + \ldots + {BW_\omega}^* \cdot {D_\omega}^* + {h_1}^* + \ldots + {h_\omega}^* \\
& \quad = (BW_1 + BW_2 + BW_3 + \ldots + BW_\omega) \cdot {D_1}^* \\
& \quad + (BW_2 + BW_3 + \ldots + BW_\omega) \cdot ({D_2}^* - {D_1}^*) \\
& \qquad\qquad \vdots \\
& \quad + (BW_\omega) \cdot ({D_\omega}^* - {D_{\omega-1}}^*) + {h_1}^* + \ldots + {h_\omega}^* \\
& \quad = (\sum_{J_j \in N} w_j) \cdot {D_1}^* \\
& \quad + (\sum_{J_j \in N} w_j - BW_1) \cdot ({D_2}^* - {D_1}^*) \\
& \qquad\qquad \vdots \\
& \quad + (\sum_{J_j \in N} w_j - BW_1 - \ldots - BW_{\omega-1}) \cdot ({D_\omega}^* - {D_{\omega-1}}^*) + {h_1}^* + \ldots + {h_\omega}^* \\
& \quad \leq (\sum_{J_j \in N} w_j) \cdot {D_1}^* \\
& \quad + (\sum_{J_j \in N} w_j - {W_1}^*) \cdot ({D_2}^* - {D_1}^*) \\
& \qquad\qquad \vdots \\
& \quad + (\sum_{J_j \in N} w_j - {W_1}^* - \ldots - {W_{\omega-1}}^*) \cdot ({D_\omega}^* - {D_{\omega-1}}^*) + {h_1}^* + \ldots + {h_\omega}^* \ \textit{due to (24)}
\end{aligned}$$

$$
\begin{aligned}
&= ({W_1}^* + {W_2}^* + {W_3}^* + \ldots + {W_\omega}^*) \cdot {D_1}^* \\
&+ ({W_2}^* + {W_3}^* + \ldots + {W_\omega}^*) \cdot ({D_2}^* - {D_1}^*) \\
&\qquad\qquad \vdots \\
&+ ({W_\omega}^*) \cdot ({D_\omega}^* - {D_{\omega-1}}^*)
\end{aligned}
$$

$$
= {W_1}^* \cdot {D_1}^* + \ldots + {W_\omega}^* \cdot {D_\omega}^* + {h_1}^* + \ldots + {h_\omega}^* = Z^*_{batch}
$$

and the proof is completed. □

Proof of Theorem 3: The result will be proved in 3 steps:

(i) We will show that $\{{x_m}^{id} : m \in M\}$ satisfies the core (in)equalities (15) when $\mathcal{W}_{\sigma_0(S)} = [a, b]$, and $\upsilon(S)$ is replaced by $\upsilon_{id}(S) = \omega_{id}([a, b])$.

(ii) We will show that $\{{x_m}^{pw} : m \in M\}$ satisfies the core (in)equalities (15) when $\mathcal{W}_{\sigma_0(S)} = [a, b]$, and $\upsilon(S)$ is replaced by $\upsilon_{pw}(S) = \omega_{pw}([a, b])$. Adding by parts would imply that $\{{x_m}^{id} + {x_m}^{pw} = {x_m}^{wip} : m \in M\}$ satisfies (15) for $\mathcal{W}_{\sigma_0(S)} = [a, b]$.

(iii) Then, we will extend parts (i) and (ii) to arbitrary window coalitions $\mathcal{W}_{\sigma_0(S)} = [a_1, b_1] \cup \ldots \cup [a_r, b_r]$.

To prove (i) note that

$$
\begin{aligned}
\sum_{m \in S} {x_m}^{id} &= \sum_{m \in S} \sum_{W_k \in \mathcal{W}_{\sigma_0(m)}} [\tfrac{1}{2} \cdot I_k \sum_{W_t \in [k+1,T] \cap \mathcal{W}_{\sigma_0}} s_t + \tfrac{1}{2} \cdot s_k \sum_{W_t \in [1,k-1] \cap \mathcal{W}_{\sigma_0}} I_t] \\
&\geq \sum_{m \in S} \sum_{W_k \in \mathcal{W}_{\sigma_0(m)}} [\tfrac{1}{2} \cdot I_k \sum_{W_t \in [k+1,T] \cap \mathcal{W}_{\sigma_0(S)}} s_t + \tfrac{1}{2} \cdot s_k \sum_{W_t \in [1,k-1] \cap \mathcal{W}_{\sigma_0(S)}} I_t] \\
&= \sum_{W_k \in \mathcal{W}_{\sigma_0(S)}} I_k \sum_{W_t \in [k+1,T] \cap \mathcal{W}_{\sigma_0(S)}} s_t \\
&= \omega_{id}([a, b]) = \upsilon_{id}(S)
\end{aligned}
$$

The second line of the above expression becomes equality when $S = M$ and hence all (in)equalities are satisfied for (M, υ_{id}).

To prove (ii) we have

$$
\begin{aligned}
\sum_{m\in S} {x_m}^{pw} &= \sum_{m\in S}\sum_{W_k\in\mathcal{W}_{\sigma_0 m}} [\lambda\cdot(\omega_{pw}([1,k])-\omega_{pw}([1,k-1]))+(1-\lambda)\cdot(\omega_{pw}([k,T])-\omega_{pw}([k+1,T]))] \\
&= \sum_{W_k\in[a,b]} \lambda\cdot(\omega_{pw}([1,k])-\omega_{pw}([1,k-1]))+(1-\lambda)\cdot(\omega_{pw}([k,T])-\omega_{pw}([k+1,T])) \\
&= \lambda\cdot(\omega_{pw}([1,a])-\omega_{pw}([1,a-1]))+(1-\lambda)\cdot(\omega_{pw}([a,T])-\omega_{pw}([a+1,T])) \\
&+\lambda\cdot(\omega_{pw}([1,a+1])-\omega_{pw}([1,a]))+(1-\lambda)\cdot(\omega_{pw}([a+1,T])-\omega_{pw}([a+2,T])) \\
&\qquad\qquad\vdots \\
&+\lambda\cdot(\omega_{pw}([1,b-1])-\omega_{pw}([1,b-2]))+(1-\lambda)\cdot(\omega_{pw}([b-1,T])-\omega_{pw}([b,T])) \\
&+\lambda\cdot(\omega_{pw}([1,b])-\omega_{pw}([1,b-1]))+(1-\lambda)\cdot(\omega_{pw}([b,T])-\omega_{pw}([b+1,T])) \\
&= \lambda\cdot(\omega_{pw}([1,b])-\omega_{pw}([1,a-1]))+(1-\lambda)\cdot(\omega_{pw}([a,T])-\omega_{pw}([b+1,T])) \\
&\geq \lambda\cdot(\omega_{pw}([a,b])+(1-\lambda)\cdot\omega_{pw}([a,b])) \\
&= \omega_{pw}([a,b])=\upsilon_{pw}(S)
\end{aligned}
$$

The inequality holds due to superadditivity of (M,v_{pw}) which can be proved similarly to (M,v_c). Note that $[1,a-1]\cup[a,b]=[1,b]$ and $[a,b]\cup[b+1,T]=[a,T]$. Therefore, $\omega_{pw}([1,a-1])+\omega_{pw}([a,b])\leq\omega_{pw}([1,b])$ and $\omega_{pw}([a,b])+\omega_{pw}([b+1,T])\leq\omega_{pw}([a,T])$. For $a=1$ and $b=T$ and assuming that $\omega_{pw}([1,0])=\omega_{pw}([T+1,T])=0$, the above expression implies:

$$\sum_{m\in M} {x_m}^{pw}=\lambda\cdot(\omega_{pw}([1,T])+(1-\lambda)\cdot\omega_{pw}([1,T]))=\omega_{pw}([1,T])=\upsilon_{pw}(M).$$

Therefore, $\{{x_1}^{pw},\ldots,{x_{|M|}}^{pw}\}$ is in the core of (M,υ_{pw}) when $\mathcal{W}_{\sigma_0(S)}=[a,b]$. Steps (i) and (ii) yield that $\{{x_1}^{wip},\ldots,{x_{|M|}}^{wip}\}$ is in the core of (M,υ_c) when $\mathcal{W}_{\sigma_0(S)}=[a,b]$. Now consider coalition S such that $\mathcal{W}_{\sigma_0(S)}=[a_1,b_1]\cup\ldots\cup[a_r,b_r]$. Admissible rearrangements should preserve the set of predecessors for jobs that belong to non-members of the coalition. For example, no interchange is allowed between jobs that complete in $[a_1,b_1]$ and jobs that complete in $[a_2,b_2]$ because this would change the set of predecessors of jobs that complete in W_{b_1+1}. Therefore, WIP savings are additive over maximally connected window coalitions. Equivalently, $\sum_{m\in S}{x_m}^{pw}\geq\sum_{k=1}^{r}\omega_{pw}([a_k,b_k])$ and $\sum_{m\in S}{x_m}^{id}\geq\sum_{k=1}^{r}\omega_{id}([a_k,b_k])$ with equalities holding when $S=M$. This proves that (22) is in the core of (M,υ_c). This completes the proof of the theorem. □

References

Aydinliyim, T. "Competition and Coordination in Outsourcing Operations", Doctoral Dissertation. Weatherhead School of Management, Case Western Reserve University, Cleveland, OH, 2007.

Borm, P., Fiestras-Janeiro, G., Hamers, H., Sánchez, E., Voorneveld, M. (2002). "On the Convexity of Games Corresponding to Sequencing Situations with Due Dates", *European Journal of Operational Research*, 136, 616-634.

Cachon, G. P. (2003). "Supply Chain Coordination with Contracts", Chapter 6, *Handbooks in Operations Research and Management Science: Supply Chain Management* edited by Graves, S., de Kok, T., North-Holland, 2003.

Cai, X. , Vairaktarakis, G. L. (2006). "Cooperative Strategies for Manufacturing Planning with Negotiable Third-Party Capacity", *Under revision*.

Calleja, P., Borm, P., Hamers, H., Klijn, F., Slikker, M. (2002). "On a New Class of Parallel Sequencing Situations and Related Games", *Annals of Operations Research*, 109, 263-276.

Calleja, P., Estévez-Fernández, M. A., Borm, P., Hamers, H. (2004). "Job Scheduling, Cooperation and Control", *CentER Discussion Papers 2004-65*, Tilburg University, Tilburg, The Netherlands, 2004.

Calleja, P., Borm, P., Hamers, H., Klijn, F., Slikker, M. (2002). "On a New Class of Parallel Sequencing Situations and Related Games", *Annals of Operations Research*, 109, 263-276.

Curiel, I., Pederzoli, G., Tijs, S. (1989). "Sequencing Games", *European Journal of Operational Research*, 40, 344-351.

Curiel, I., Potters, J., Rajendra Prasad, V., Tijs, S., Veltman, B. (1994). "Sequencing and Cooperation", *Operations Research*, 42, 566-568.

Curiel, I., Hamers, H., Klijn, F. (2002). "Sequencing Games: A Survey", *in P. Borm and H. Peters, eds. 'Chapters in Game Theory: In Honor of Stef Tijs'* Kluwer Academic Publishers, Boston, pp. 27-50.

Elmaghraby, W. J. (2000). "Supply Contract Competition and Sourcing Policies", *Manufacturing and Service Operations Management*, 2, 350-371.

Grosvenor, F., Austin, T.A. (2001). "Cisco's eHub Initiative", *Supply Chain Management Review*, July/August, 28-35, 2001.

Hamers, H., Borm, P., Tijs, S. (1995). "On Games Corresponding to Sequencing Situations with Ready Times", *Mathematical Programming*, 70, 1-13.

Hamers, H., Klijn, F., Suijs, J. (1999). "On the Balancedness of Multimachine Sequencing Games", *European Journal of Operational Research*, 119, 678-691.

Hamers, H., Klijn, F., van Velzen, B. (2002). "On the Convexity Precedence Sequencing Games", *CentER Discussion Papers 2002-112*, Tilburg University, Tilburg, The Netherlands, 2002.

Papadimitriou, C. (2001). "Algorithms, Games and the Internet", *Proceedings of 33rd STOC*, 2001, 749-753.

Rosenthal, R. W. (1973). "A Class of Games Possessing Pure-Strategy Nash Equilibria", *International Journal of Game Theory*, 2, 65-67.

Sandholm, W. H. (2002). "Evolutionary Implementation and Congestion Pricing", *The Review of Economic Studies*, 69, 667-689.

Shapley, L. (1971). "Cores of Convex Games", *International Journal of Game Theory*, 1, 11-26.

Smith, W. (1956) "Various optimizers for single-stage production." *Naval Research Logistics Quarterly*, 3, 59-66.

Wellman, M. P., Walsh, W. E. (2001). "Auction Protocols for Decentralized Scheduling", *Games and Economic Behavior*, 35, 271-303.