DISTRIBUTED OPTIMIZATION UNDER PARTIAL INFORMATION USING DIRECT INTERACTION: A METHODOLOGY AND APPLICATIONS

A Dissertation

by

SUN WOO KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2005

Major Subject: Industrial Engineering

DISTRIBUTED OPTIMIZATION UNDER PARTIAL INFORMATION USING DIRECT INTERACTION: A METHODOLOGY AND APPLICATIONS

A Dissertation

by

SUN WOO KIM

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee, V. Jorge Leon
Committee Members, Cesar O. Malave
Amarnath Banerjee
Sheng-Jen Hsieh
Head of Department, Brett A. Peters

December 2005

Major Subject: Industrial Engineering

ABSTRACT

Distributed Optimization Under Partial Information Using Direct Interaction: A Methodology and Applications. (December 2005)

Sun Woo Kim, B.S., Yonsei University, Korea;

M.S., Yonsei University, Korea

Chair of Advisory Committee: Dr. V. Jorge Leon

This research proposes a methodology to solve distributed optimization problems where quasi-autonomous decision entities directly interact with each other for partial information sharing. In the distributed system we study the quasi-autonomy arising from the assumption that each decision entity has complete and unique responsibility for a subset of decision variables. However, when solving a decision problem locally, consideration is given to how the local decisions affect overall system performance such that close-to-optimal solutions are obtained among all participating decision entities. Partial information sharing refers to the fact that no entity has the complete information access needed to solve the optimization problem globally. This condition hinders the direct application of traditional optimization solution methods. In this research, it is further assumed that direct interaction among the decision entities is allowed. This compensates for the lack of complete information access with the interactive exchange of non-private information. The methodology is tested in different application contexts: manufacturing capacity allocation, single machine scheduling, and jobshop scheduling. The experimental results show that the proposed method generates close-to optimal solutions in the tested problem settings.

To my family

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. V. Jorge Leon who helped me whenever I was in academic difficulty and encouraged me to complete the Ph. D work. He himself showed me how I should behave as a professional researcher. I also appreciate Dr. Cesar O. Malave, Dr. Amarnath Banerjee and Dr. Sheng-Jen Hsieh for serving as my committee and giving valuable suggestions.

I sincerely thank my parents, Jong Yul Kim and Jae Soon Gou, my parents-in-law Haeng Won Chung and Seo Jung Choi, and all other family in Korea, for their endless support and encouragement.

Finally, I would like to dedicate all of my work to my wife So Young Chung, my son June-Suh Kim, and my daughter Yea-Rin Kim. Without their love, faith and sacrifice, I would never accomplish this work. They deserve all of mine and I love them forever.

TABLE OF CONTENTS

Page

CHAPTER Page

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

1. Overview of the research

This research proposes a methodology to solve a class of distributed optimization problems where quasi-autonomous decision entities directly interact with each other only for partial information sharing. In the distributed system we study, the quasi-autonomy arises from the assumption that each decision entity has responsibility for a subset of decision variables. However, when solving a decision problem locally, consideration is given to how the local decisions affect overall system performance. This ensures that close-to-optimal solutions are obtained among all participating decision entities. Partial information sharing occurs when no entity has the complete information access needed to solve the optimization problem globally. This condition hinders the direct application of traditional optimization solutions methods. In this research, it is further assumed that direct interaction among the decision entities is allowed. This compensates for the lack of complete information access with the interactive exchange of non-private information. This approach extends the work by Jeong and Leon (2002) where intermediary 'coupling agents' were used between interacting descision entities to solve the distrubuted optimization problem. The methodology is tested in different application contexts: manufacturing capacity allocation, single machine scheduling, and jobshop scheduling.

2. Motivation

Consider a product and the company that manufactures the product. In the manufacturing company, there are several departments which are responsible for unique processes required to make the product. Each department has a manager to take charge

---

This dissertation follows the format and style of *IIE Transactions*.

of the tasks and the decisions in it. Ultimately, these departments cooperate for the company's goal, such as maximizing total profit. However, interdepartmental conflicts can arise due to the heterogeneousness of the tasks and the associated viewpoints of the product features. These conflicts can be resolved by sharing information and opinions without restriction or by a supervising authority with a global view of the company. But what if the information sharing is limited and no central coordination is possible?

These days, subcontracting is a popular cost-saving measure allowing companies to concentrate on core processes, while subcontractors deal with minor processes. Take a look at the subcontracting method. How are conflicts between departments in the company and the subcontractors handled? The problem solving procedures will be similar: through the interactions between the departments and the subcontractors, they will find a compromised resolution. But there are two important things to keep in mind: restricted information sharing and the difficulty of centralized coordination. Even though the subcontractors cooperate for the company's goal, they are independent systems. Exposing their private information to others may risk their autonomy. Therefore, the subcontractors will provide only the information necessary to settle the conflicts. And the supervising authority of the company can hardly intervene in the decision making procedure of the subcontractors and control them. Therefore, centralized modeling and solution methods are not applicable to the decision making and the associated optimization problems in this case.

It is not difficult to find real-world examples similar to the second scenario. Consider an automotive manufacturing supply chain model: modular consortium (Ure and Jaegersberg, 2005). In a modular consortium, all suppliers are located in one site as the main automotive manufacturer. They assemble the modular parts for cars on their own production lines, then delivering these parts into the main production lines *Just in Time (JIT)* and *Just in Sequence (JIS)*. Thus, the production schedules between the suppliers and the main manufacturer are closely connected. For example, if scheduling conflict problems occurred, they can be better treated by a distributed method.

The organizational structure of NIKE may require a distributed approach. NIKE owns only design and marketing departments, outsourcing the manufacturing facilities in countries for production. The decision making problems under this structure will be another example that requires a distributed methods. Similar cases can be found from the companies using OEM type production.

In this research, we propose a distributed modeling and solution methodology to effectively deal with the decision making and optimization problems that arise in the aforementioned situations.

3. Problem statement

In this section, we present the Distributed Optimization Under Direct Interaction and Partial Information Sharing Problem (DIPIP).

A distributed system is comprised of multiple subsystems. Each subsystem has a quasi-autonomous decision maker who takes full control of the local problem solving and distinctive information associated with its own task and goal. Similar to Jeong and Leon (2002), a vector of decision variables, $\boldsymbol{x}_i$, is associated with the local decision variables of subsystem i, where $i = 1,...,m$. Each subsystem i has a local objective, $f_i(\boldsymbol{x}_i)$, subject to local constraints, $B_i\boldsymbol{x}_i \le \boldsymbol{b}_i$; where, $\boldsymbol{x}_i$ is $1 \times u_i$ vector, where u_i is the number of decision variables of the problem, and b_i is $1 \times v_i$, where v_i is the number of local constraints of the problem.

All the subsystems are directly or indirectly interconnected by coupling constraints. Let $\boldsymbol{x}_{ij}$, $x_{ij} \subset x_i$, be the set of local decision variables of subsystem i that are coupled with other subsystems in the j^{th} coupling constraint. S_j is the set of subsystems that are coupled in j^{th} coupling constraint. Subsystems i and k are coupled if they are in the same S_j for some j. T_i is the index-set of the coupling constraints where subsystem i is coupled with other subsystems. The vector of non-coupling variables $\boldsymbol{x}_{i0}$ is comprised of the local decision variables of subsystem i that are not related in any coupling constraint. All local variables are mutually exclusive among subsystems; that is $\boldsymbol{x}_i \neq \boldsymbol{x}_k$ for all $i \neq k$, $i, k \in U$ where U is the index-set of the subsystems that consist of the global distributed

system and participate in the global system optimization procedure. The coupling constraints are represented as $\sum_{i \in S_j} A_{ij} x_{ij} \leq a_j,\ j = 1,...,r$; where r is the number of coupling constraints in the distributed system.

The overall distributed optimization problem can be written as follows:

(DIPIP): (1.1)

$$\text{Opt} \sum_{i=1}^{m} f_i(x_i) \quad \text{(i)}$$

s.t.

All local constraints: $B_i x_i \leq b_i,\ i = 1,...,m$ (ii)

All coupling constraints: $\sum_{i \in S_j} A_{ij} x_{ij} \leq a_j,\ j = 1,...,r$ (iii)

Quasi-autonomy condition (iv)

Partial information sharing condition (v)

Direct interaction condition (vi)

The objective function in (i) is the sum of all the local objectives. The goal of this research is to pursue globally optimal and feasible solutions while local subsystem problems are solved independently by quasi-autonomous decision makers. The global system goal is the minimization of the sum of deviations in the subsystem solutions from the global optimal solution. In a minimization problem the global objective can be written as:

$$\min \sum_{i=1}^{m} (f_i(x_i) - f_i(x_i^*)) = \min \sum_{i=1}^{m} f_i(x_i) \quad (1.2)$$

Constraint (ii) states that all local constraints must be satisfied.

Constraint (iii) states that all coupling constraints must be satisfied. In this research, it is assumed that $|T_i| \geq 1$ for $i \in U$ and $|S_j| \geq 2$ for all $j \in T_i$. $|T_i| \geq 1$ means that subsystem i is coupled with other subsystems in at least one coupling constraint. If $|S_j| = 1$ and the corresponding subsystem index is i, then j^{th} coupling constraint can be represented only by the local decision variables of the subsystem i. It would thus be considered a local constraint of subsystem i; that is, $A_j \mathbf{x_i} \leq a_j$. Therefore, the assumption

implies that every subsystem is coupled with at least one other subsystem in one or more coupling constraints. Note that Figure 1 shows the block angular structure of the problem in a mathematical formulation.

Conditions (iv), (v) and (vi) make DIPIP a novel distributed optimization problem under partial information sharing:

- The quasi-autonomy condition in (iv) requires that only subsystem i has the authority to specify the values of the decision variables $\boldsymbol{x}_i$
- The partial information sharing condition in (v) specifies what information is private and what information can be shared with others:
  a) The local objective, $f_i(x_i)$, and the local constraints, $B_i\boldsymbol{x_i} \leq b_i$, are private information of subsystem i.
  b) The values of $\boldsymbol{x}_i$ associated with a given coupling constraint can be shared with the corresponding coupled subsystems.
  c) Subsystem i can partially view all the coupling contraints that contain any variable in $\boldsymbol{x}_i$. That is, in the j^{th} coupling constraint, subsystem i can only view the coefficient vector, A_{ij}, of its own decision variable vector and $\boldsymbol{a_j}$. This is an important difference from CICA (Jeong and Leon, 2002) where coupling agents have complete information about the coupling constraints and coupled autonomous organizations only recognize coupling agents with which they should interact.
  d) Subsystem i knows what other subsystems it is associated in coupling constraints; i.e. S_j is known to all the subsystems that are coupled in j^{th} coupling constraint. This is also an important difference with the problem studied by Jeong and Leon (2002), where "who" were the other subsystems was not explicitly known.
- Direct interaction condition in (vi): Coupled subsystems i and k must interact directly without mediation by passing non-private information vectors between each other. This interaction must follow a pre-specified interaction protocol. This is

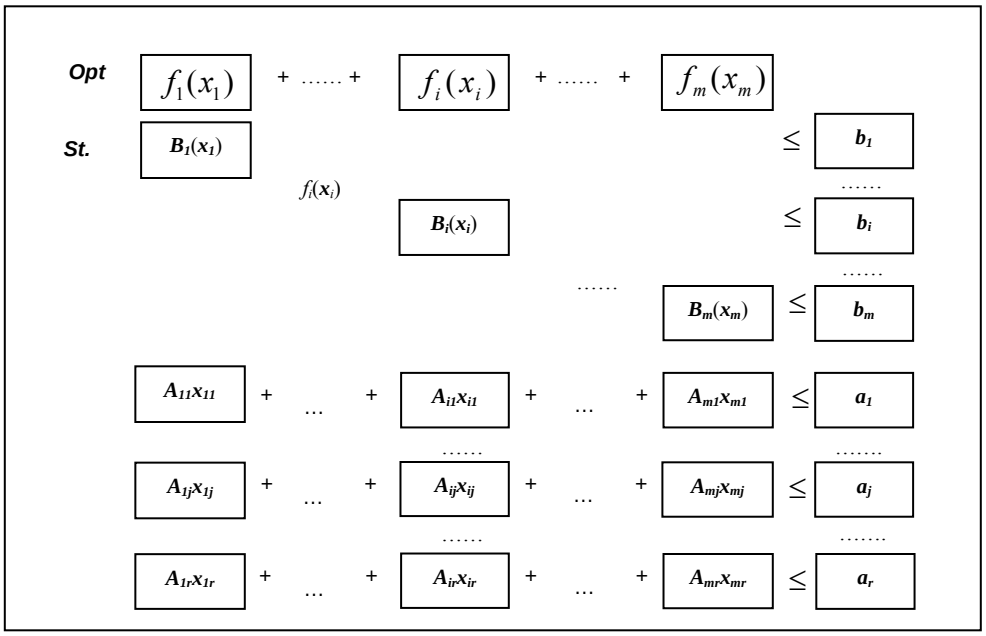

Figure 1 The block angular structure of a distributed system.

another difference with the work by Jeong and Leon (2002) where the interaction is through 'coupling agents'.

4. Research issues

As described in the previous section, the DIPIP problem in (1.1) has unique characteristics that hinder the direct application of traditional optimization methods. Especially challenging is the fact that no one entity has full access to all the information required to solve the problem globally. When solving DIPIP, each subsystem can only view and control its local subproblem and the information that is received from other subsystems each iteration during the solution phase. Hence, it is necessary to (1) define the appropriate information sharing between interacting subsystems, (2) formulate the local subproblems such that private information is not shared, and (3) design a suitable interaction protocol.

4.1. *Information sharing*

Given two interacting subsystems, we assume that the minimum information that must be exchanged includes (1) the desired solutions, and (2) information about the local objective losses and local constraint violations for deviations from the solution in (1). It is assumed that the information vector from a subsystem to another subsystem is composed of the local solution vector and a penalty weight vector ($\overline{y_{ij}}, \alpha_{ij}$). The local solution $\overline{y_{ij}}$ satisfies local constraints and coupling constraints in T_i locally. The penalty weight α_{ij} reflects the marginal variation of the local objective value and possible constraint violations if one deviates from the solution $\overline{y_{ij}}$. When designing α_{ij} , care must be taken not to disclose information that is private to subsystem *i*. More details on the determination of α_{ij} are included Chapter II.

4.2. *Local subproblem*

DIPIP requires that subsystem *i* locally solves a subproblem, **SUB$_i$**. By carefully relaxing the coupling constraints to the objective function, each subsystem problem can be formulated independently but must include an augmented objective function f(.)+g(.). In this function, g_i(.) must reflect the objective and constraints of subsystems with which it is coupled. Given the information vectors from coupled subsystems, $\left(\overline{y_{ij}}, \alpha_{ij}\right)$, where *j* is coupled with *i*:

$$\textbf{SUB}_i\text{: } (\text{OPT } f_i(x_i) + g_i(x_{ij} \mid \overline{y_{ij}}, \alpha_{ij}\beta_{ij}, \forall k \in S_j, k \neq i \ ) \text{, s.t. } B_i x_i \leq b_i) \qquad (1.3)$$

A detailed derivation of g(.) is given in Chapter II.

4.3. *Interaction protocol*

An interaction protocol must be designed to be suitable for partial information sharing and also make direct interactions possible without third party mediation. There are two types of interaction methods: synchronous interaction, in which all the subsystems

exchange the information vectors simultaneously by the predetermined schedules; and asynchronous interaction, in which interactions occur independently between pairs of subsystems.

The most important advantage of synchronous interactions is to get the information from all other participating decision entities at the same time. This helps the local solutions be updated to reflect all other subsystems' desire in a balanced and timely manner. However, it is a rigid type of interaction method. For example, as the number of participating entities increases, it will become difficult to make all the interactions synchronized.

On the other hand, asynchronous interactions is flexible in that any participating entity can initiate the interaction. In other words, local decision entities can interact with others whenever they need updated information in their schedule. But it also implies that they may not get the updated information from all other participants instantly. This can cause biased local solutions from the updated information only. Another potential disadvantage is that it may take more time in achieving a compromised solution, with a higher possibility of oscillating local solutions as well.

5. Literature survey

In this section, we review previous studies on distributed methods for large-scale optimization problems. Even though a number of heuristic-based methods exist for distributed optimization problem solving, the main objective of this research is to develop a distributed methodology with a strong mathematical programming basis. Therefore, we concentrate our review on related literature associated with mathematical programming. The distributed methods without mathematical programming background are mentioned last.

5.1. *Centralized decomposition method for large-scale optimization problems*

To efficiently solve large-scale optimization problems, mathematical decomposition techniques have been studied in the Operations Research area. The main idea behind

those methods is to decompose a large problem into multiple small-size problems, such as a master problem and subproblems in the Dantzig-Wolfe decomposition method. The master problem manages the shared resources among multiple subproblems by either updating the prices of the resources (Dantzig and Wolfe, 1961) or allocating the resources to subproblems (Bender, 1962; Kate, 1972). Subproblems are then solved to update either their demand for the resources or their desired price of the resources. The former approach is known as price-directed and the latter as resource-directed, classified by the information type the master problem sends to the subproblems. Lagrangian relaxation (Fisher, 1981) is also a popular technique explored by many OR researchers as a basis of distributed solution methods for large optimization problems (Roundy *et al.,* 1991). However, in the methods mentioned above, a single decision maker has full access to the complete problem formulation which is not allowed by constraints (iv)-(vi) in formulation (1.1).

5.2. *Distributed algorithm with unrestricted information sharing*

There are distributed algorithms that, for problem solving, allow any global information to be shared without restriction. Gou *et al.* (1994) applied the holonic system approach to a robotic assembly planning problem and introduced a distributed algorithm based upon Lagrangian relaxation concepts. The proposed model includes a central coordinator that collects all the necessary, though not complete, information to calculate coordination parameter values. Murphy *et al.* (1995) studied a decomposition-coordination approach for large-scale optimization problems. In their research, a large problem is decomposed into several subproblems by duplicating coupled decision variables that they call boundary variables. Then, the Auxiliary Problem Principle is applied to minimize the Augmented Lagrangian subproblems. Although subproblems are solved independently, the addition of the consistency constraints maintains the same boundary variable values for all subproblems. Again, any necessary information can be exchanged to solve subproblems as the process repeats.

5.3. *Distributed algorithm with restricted information sharing*

Since auctions can intuitively be viewed as a distributed process, a significant amount of work has been done to apply the auction mechanism to various problems. There are two main entities in the auction: the auctioneer, who collects bidding information from multiple bidders, updates resource prices, and allocates the resources to bidders; and the bidder, who bids for the limited resources. The exchanged information between the auctioneer and the bidder is restricted by a predetermined interaction mechanism. Bertsekas proposed an auction algorithm for assignment problems of linear programming (Bertsekas, 1988). In this study, he modeled the assignment problem as a multiple-person multiple-object auction. ε - Complementary Slackness (ε - CS) was introduced to avoid possible cyclical behaviors and to facilitate the algorithm in optimization contexts. The algorithm was later extended to transportation problems and other linear network optimization problems (Bertsekas and Tsitsiklis, 1989; Bertsekas, 1990). Kutanoglu and Wu (1999) explored a group of combinatorial auction mechanisms for resource scheduling problems and showed that Lagrangian relaxation is equivalent to one of the combinatorial auction mechanisms. Ertogral and Wu (2000) applied the auction mechanism to solve production planning problems for multiple facilities. Guo *et al.* (2002) developed a market-oriented decomposition method based on double auctions for large-scale linear programming problems. They illustrated the relationship between the auction algorithm and mathematical programming by exploring similarities and differences between the proposed market-based algorithm and Dantzig-Wolfe decomposition method with economic interpretations.

As a distributed method of much stronger mathematical programming basis, Jeong and Leon (2002) proposed a distributed methodology called CICA. In the CICA approach, two main entities are introduced, CAO and CA, to model the distributed problems. CAOs are quasi-autonomous organizations that solve their own local problems with independent decision authority. However, multiple CAOs are interrelated by coupling constraints. CAs manage the coupling constraints to help CAOs achieve compromised solutions that do not violate coupling constraints, as well as local

constraints. Both CAOs and CA do not have complete system information. While applied to machine scheduling problems and capacity allocation problems, only partial information can be shared during interactions between each CAO and CA.

5.4. *Other distributed algorithms*

In this section reviews distributed methods based on heuristics or economic theory without a mathematical programming formulation.

Originating from the distributed AI approach, the distributed constraint heuristic search has been popularly studied for machine scheduling problems. In this approach, job scheduling can be done on multiagent environments with a job-based perspective, resource-based perspective, or a combination of the two. The Intelligent Scheduling and Information System (ISIS) (Fox and Smith, 1984) and Opportunistic Intelligent Scheduling (OPIS) (Smith *et al.*, 1990) were developed for factory scheduling. The former has adopted the job-based perspective, and the latter the combined perspective. Information sharing is unrestricted for both systems. Sycara *et al.* (1991) studied a micro-opportunistic scheduling system, based on the decomposition of resource periods and the allocation of the operation according to the bottleneck period. In their method, similar to CAs in CICA, third party *monitoring agents* exist to manage shared resources. Only partial information is shared, but any local information can be exchanged if other agents desire.

Talukda *et al.* (1998) also proposed a distributed problem solving method called Asynchronous-team (A-team) based on multiagent systems. In their method, multiple autonomous agents with unique problem-solving techniques work together to solve large problems cooperatively. Each agent solves the problem independently, but using up-to-date partial solutions of other agents without restriction.

Another important stream of distributed problem solving studies is the market-like approach that attempted to interpret manufacturing control system behavior using economic theories. *Contract-net*, the interaction protocol used to allocate tasks to agents in distributed environments was first proposed by Davis and Smith (1983). It has been

applied for dynamic scheduling in manufacturing control systems (Duffie and Prabhu, 1994; Lin and Solberg, 1994). For static scheduling, Wellman *et al.* (2001) developed auction protocols for decentralized scheduling problems. The proposed market mechanisms uses price information, derived via distributed bidding protocols, to determine schedules. All of these methods are based on interactions between bidding agents and resource agents to exchange restricted information.

6. Research contributions

The main research contributions of this dissertation are:

(i) Quasi-autonomy and partial information: We develop a methodology for modeling the distributed system and solving the associated optimization problems. In the system under study, no decision entity has the global view of the entire system or complete system information. While quasi-autonomous local decision entities solve the local problems indpendently, they cooperate to find a close-to-optimal system solution. To compensate for the lack of system information, partial local information sharing is allowed. Therefore, the proposed method is designed to search for the system solution under partial information sharing among quasi-autonomous decision entities.

(ii) Direct interactions: The proposed method will be based on direct interactions for cooperative partial information sharing among local decision entities. Neither central coordination nor third party mediation is necessary in the proposed method. For example, the master problem in Lagrangian Relaxation and coupling agents (CAs) in CICA take the role of the coordination of local solutions in global preference. There is no previous research about direct interactions for information sharing in a distributed system.

(iii) Interaction protocol for partial information sharing: The interaction protocol will be designed to exchange implicit but essential information sufficient to pursue the global goal without complete system information or global coordination.

(iv) Mathematical programming basis: We develop a solution methodology that can be applied to the optimization problems formulated by linear programming and integer programming. An LP heuristic is proposed to deal with the problems of large sizes in practical times.

7. Organization of the dissertation

This dissertation is organized as follows: Chapter II introduces a general distributed model and proposes a solution methodology, Direct Interaction for Partial Information Sharing (DIPIS). In Chapter III, We study the distributed capacity allocation problem in a linear programming model. The proposed method is applied to single-machine scheduling problems in Chapter IV. In Chapter V, jobshop scheduling problems are investigated for the applicability of the proposed method to more complicated models. Finally, Chapter VI summarizes the research contributions and the directions of future study.

CHAPTER II

METHODOLOGY FOR DISTRIBUTED OPTIMIZATION USING DIRECT INTERACTION UNDER PARTIAL INFORMATION SHARING

In this chapter we develop the Direct Interaction for Partial Information Sharing (DIPIS) method to solve (DIPIP) described in Chapter I. DIPIS decomposes DIPIP into interdependent subproblems using Lagrangean Relaxation concepts. These concepts are modified to ensure that private information remains local, and that no master problem is required to update the Lagrangian multipliers. At each iteration in DIPIS the subsystems solve local subproblems, computes a penalty vector (that reflects objective function and constraint violation if the current solutin is modified), and then pass the current solution and penalty weights to corresponding coupled subsystems. This chapter starts by deriving the subsystem problem, then it describes the derivation of the penalty weights. It also describes the penalty updating from one iteration to the next used to reflect the new information from the other subsystems. Finally it presents the complete algorithm.

1. Subsystem problem

Observing the subsystem problem **(SUB$_i$)** in expression (1.3) the main challenge is to derive an appropriate augmenting function $g(.)$. If complete system information is available, the objective function of subsystem i is as follow (Jeong and Leon, 2002):

$$f_i(x_i) + \sum_{j \in T_i} g_j(x_{kj}, k \in S_j) = f_i(x_i) \pm \sum_{j \in T_i} (\theta_{ij}(a_j - \sum_{k \in S_j} A_{kj} x_{kj})) \qquad (2.1)$$

In addition to the original objective function, the penalty functions related to the relaxed constraints must be included in the subsystem optimization problem for the pursuit of the global goal. θ_{ij}, a vector of positive Lagrangian multipliers in the penalty function, can be calculated and updated using the system information without restriction. Hence, the violations of the subsystems' solutions on the coupling constraints will be appropriately penalized. In the distributed system problem only partial information

sharing is allowed among subsystems and therefore the exact penalty functions cannot be generated. Specifically, in (2.1) the coefficients of the decision variables of other subsystems in the coupling constraints, $(A_{kj}, \forall k \in S_j - \{i\})$, are unknown to the subsystem *i*. And the decision variables of other subsystems, $(x_{kj}, \forall k \in S_j - \{i\})$, cannot be controlled by subsystem *i*. Consequently, $\sum_{j \in T_i} g_j(x_{kj}, k \in S_j)$ must be approximated using only the information vectors from other subsystems, $\{(\overline{y}_{kj}, \alpha_{kj}),\ \forall k \in S_j - \{i\}\}$. This approximation makes the penalty functions represented by the decision variables of subsystem *i*. The resulting functions still have to reflect other subsystems' desires in the coupling constraints so that the subsystem can find a compromised solution in global preference. Detailed derivation of the function approximation will be explained next.

The penalty functions in (2.1) can be rearranged as follows,

$$\sum_{j \in T_i} g_j(x_{kj}, k \in S_j) = \sum_{j \in T_i} (\theta_{ij}(a_j - A_{ij}x_{ij} - \sum_{k \in S_j - \{i\}} A_{kj}x_{kj})) \tag{2.2}$$

Again, $\sum_{k \in S_j - \{i\}} A_{kj}\, x_{kj}$ is unknown to subsystem *i*. Let $y_{kj} = A_{kj}\, x_{kj}$ for $\forall\ k \in S_j - \{i\}$ and then the penalty function can be restated as,

$$\sum_{j \in T_i} g_j(x_{ij}, y_{kj}, \forall k \in S_j - \{i\}) = \sum_{j \in T_i} (\theta_{ij}(a_j - A_{ij}x_{ij} - \sum_{k \in S_j - \{i\}} y_{kj})) \tag{2.3}$$

Consider an element of the penalty function associated with j^{th} coupling constraint, $g_j(x_{ij}, y_{kj}, \forall k \in S_j - \{i\})$. Here subsystem *i* has no control of y_{kj} , the decision variables of subsystem *k*. At the end of $(n\text{-}1)^{\text{th}}$ iteration, subsystem *i* receives partial information $(\overline{y}_{kj}, \alpha_{kj})$ from subsystem *k* where $\forall k \in S_j - \{i\}$. Then, by substituting $\overline{y}_{kj}$ for y_{kj} , the penalty function can now be represented by only x_{ij} , the decision variables of subsystem *i*. The cooperative solution $\overline{x}_{ij}$ for x_{ij} of subsystem *i* can be easily calculated using the coupling constraint with the information as follows,

$$\overline{x}_{ij} = A_{ij}^{-1} \cdot \overline{r}_j \quad \text{where } \overline{r}_j = \max(0, (a_j - \sum_{k \in S_j - \{i\}} \overline{y}_{kj})) \tag{2.4}$$

$\bar{x}_{ij}$ represents maximum avaiable value that x_{ij} can get regarding j^{th} coupling constraint, given $\bar{y}_{kj}$ for $\forall k \in S_j - \{i\}$.

Similar to the derivation steps used in CICA model (Jeong and Leon, 2002), applying the first order Taylor expansion for the approximation of $g_j(x_{ij})$ at $x_{ij} = \bar{x}_{ij}$, we get,

$$g_j(x_{ij}) = g_j(\bar{x}_{ij}) + \nabla g_j(\bar{x}_{ij})(x_{ij} - \bar{x}_{ij}) \tag{2.5}$$

where $\nabla g_j(\bar{x}_{ij}) = \dfrac{\partial g_j(\bar{r}_j)}{\partial x_{ij}} = \sum_{k \in S_j - \{i\}} \left(\dfrac{\partial g_j(\bar{y}_{kj}, \forall k \in S_j - \{i\})}{\partial y_{kj}} \right) \left(\dfrac{\partial y_{kj}}{\partial x_{ij}} \right)$.

Since $\alpha_{kj} = \dfrac{\partial g_j(\bar{y}_{kj}, \forall k \in S_j - \{i\})}{\partial y_{kj}}$ and $\dfrac{\partial y_{kj}}{\partial x_{ij}} = -A_{ij}$, (2.5) is equivalent to the following equation,

$$g_j(x_{ij}) = g_j(\bar{x}_{ij}) - (A_{ij} \sum_{k \in S_j - \{i\}} \alpha_{kj})(x_{ij} - \bar{x}_{ij}) \tag{2.6}$$

Using (2.4), $\bar{x}_{ij}$ can be replaced back by $\bar{r}_j$. And $g_j(\bar{x}_{ij})$ is a constant term and thus can be excluded in the optimization process. The concluding formulation of $g_j(x_{ij})$ is,

$$g_j(x_{ij}) = (\sum_{k \in S_j - \{i\}} \alpha_{kj})(A_{ij} x_{ij} - \bar{r}_j) \tag{2.7}$$

Hence, the subsystem problem can be formulated as follows:

$$\text{SUB}_i\text{: (OPT } f_i(x_i) \pm \sum_{j \in T_i} \left((\sum_{k \in S_j - \{i\}} \alpha_{kj})(A_{ij} x_{ij} - r_j) \right), \text{s.t. } B_i x_i \le b_i) \tag{2.8}$$

$$\text{where } r_j = \max(0, (a_j - \sum_{k \in S_j - \{i\}} \bar{y}_{kj}))$$

The plus sign in the objective function is for minimization problems and the minus sign is for maximization problems.

2. Derivation of the penalty weights

Given the information above, for a differentiable function g_j, α_{kj}, the penalty weight received from subsystem *k* associated with j^{th} coupling constraint, can be defined as follows,

$$\alpha_{kj} = \left. \frac{\partial g_j(y_{kj}, \bar{y}_{lj}, \forall l \in S_j - \{k\})}{\partial y_{kj}} \right|_{y_{kj} = \bar{y}_{kj} - \Delta} \tag{2.9}$$

That is, α_{kj} implies the marginal variation of the function g_j, as the value of y_{kj} decreases from $\bar{y}_{kj}$ by small positive constant Δ with values of all other variables fixed as given.

If g_j is not differentiable, α_{kj} can be calculated by the form,

$$\alpha_{kj} = \frac{g_j(\bar{y}_{kj}, \bar{y}_{lj}, \forall l \in S_j - \{k\}) - g_j(\bar{y}_{kj} - \Delta, \bar{y}_{lj}, \forall l \in S_j - \{k\})}{\Delta} \tag{2.10}$$

(2.9) and (2.10) do not require complete system information related to the coupling constraints. Instead, they can be determined by using the partial information supplied by other subsystems.

3. Lagrangian multiplier update for SUB$_i$

Section 3 describes a Lagrangian multiplier update procedure in DIPIS. The proposed procedure is based on the traditional subgradient optimization method, modified to comply with the partial information restrictions present in DIPIP.

At the end of $(n\text{-}1)^{\text{th}}$ iteration, the subsystems interrelated with SUB$_i$ by coupling constraints send the associated system information $\{(\bar{y}_{kj}^{n-1}, \alpha_{kj}^{n-1}),\ k \in S_j - \{i\} \}$ to subsystem *i*. Then, at n^{th} iteration, subsystem *i* solves the following problem,

$$\text{SUB}_i\text{: (OPT } f_i(x_i^n) + \sum_{j \in T_i} \left(\left(\sum_{k \in S_j - \{i\}} \alpha_{kj}^{n-1} \right)(A_{ij} x_{ij}^n - \bar{r}_j^n) \right), \text{s.t. } B_i x_i^n \le b_i) \tag{2.11}$$

$$\text{where } \bar{r}_j^n = \max(0, (a_j - \sum_{k \in S_j - \{i\}} \bar{y}_{kj}^{n-1}))$$

Let $x_i^{n*} = (x_{i0}^{n*}, x_{ij}^{n*})$ be the solution of (2.14). Given $\bar{y}_{ki}^{n-1}$, the cooperative solution $\bar{x}_{ij}^n$ is obtained using (2.4). Then, θ_{ij}^n can be updated as follows:1

$$\theta_{ij}^{\mathbf{n}} = \max(0, \theta_{ij}^{\mathbf{n}-1} - s_i^n (\bar{r}_j^n - A_{ij} x_{ij}^{n*})) \tag{2.12}$$

$$s_j^n = \frac{\tau_j^n \left| f_i(x_i^{n*}) - f_i(x_{i0}^{n*}, \hat{x}_{ij}^n) \right|}{\sum_{j \in T_i} (\bar{r}_j^n - A_{ij} x_{ij}^{n*})^2} \quad \text{where } \hat{x}_{ij}^n \le \bar{x}_{ij}^n \tag{2.13}$$

$$\tau_j^n = \tau_j^{n-1} \times p, \text{ if } \left| f_i(x_i^{n*}) - f_i(x_{i0}^{n*}, \hat{x}_{ij}^n) \right| > \left| f_i(x_i^{n-1*}) - f_i(x_{i0}^{n-1*}, \hat{x}_{ij}^{n-1}) \right|$$

$$= \tau_j^{n-1}, \text{ otherwise} \tag{2.14}$$

where s_j^n is a positive step size, τ_j^n is a positive step length, p is the step parameter with $0 < p < 1$ and $f_i(x_i^{n*})$ is the current objective value of Sub$_i$. $f_i(x_{i0}^{n*}, \hat{x}_{ij}^n)$ is the objective value when Sub$_i$ problem is solved to minimize the conflict from the solutions of the coupled subsystems at $(n\text{-}1)^{\text{th}}$ iteration.

The updating method is based on the subgradient optimization technique that is one of the most popular methods for solve Lagrangian Relaxation problems. The main difference of the proposed method from the traditional subgradient optimization technique is that the step size can be updated using partial information sent by coupled subsystems. Traditional Lagrangian multiplier updating rules require a globally feasible upperbound solution which can be calculated with complete system information. In the proposed model, complete system information is not available to any local subsystem, making calculation of the centrally coordinated Lagrangian multipliers impossible. The proposed method updates Lagrangian multipliers locally, but appropriately; it uses partial system information obtained via direct interactions among the associated subsystems to guide the local solutions towards a compromised state. That is, it works to minimize conflict among local solutions associated with the coupling constraints.

4. DIPIS algorithm

The steps of DIPIS algorithm in solving a distributed problem are as follows:

Initialization: Set the number of maximum interaction *N*.

Set $\alpha_{kj}^{0} = 0$, $\bar{y}_{kj}^{0} = 0$, $s_j^0 = 0$, $\tau_j^0 = 2$ and p, $0 < p < 1$.

Set $n = 1$.

Step 1: SUB$_i$ problem. For all $i = 1,..,m$.

Step 1.1. Solve subsystem problem **SUB$_i$**. in (2.11).

Step 1.2. Update step length τ_i^n as shown in (2.14).

Step 1.3. Update step size s_i^n as shown in (2.13).

Step 1.4. Update θ_{ij}^n as shown in (2.12).

Step 1.5 Calculate α_{ij}^n as shown in (2.9) or (2.10)

Step 1.6. Distribute α_{ij}^n and $\bar{y}_{ij}^n$ to the coupled subsystems k, $\forall k \in S_j - \{i\}$.

Step 2: If any of the predetermined stopping criteria is satisfied, stop. Otherwise $n = n + 1$ and go to **Step 1**.

The solution procedure starts with the initialization: setting all the penalty weights equal to zero and other parameters for Lagrangian multiplier updating to starting values. At every iteration, each subsystem solves its local problem and updates the penalty weights, as shown in step1. For the penalty weight updates, the subsystems need to get a cooperative solution which minimizes the conflict with other subsystems. The cooperative solution of subsystem *i* at n^{th} iteration can be calculated as follows:

(i) Substitute the solutions of other subsystms received at the end of the previous iteration, $\bar{y}_{kj}^{n-1}$, $\forall k \in S_j - \{i\}, j$, into the coupling constraints and calculate $\bar{x}_{ij}^n$ using (2.4) .

(ii) Then, solve the local problem by applying additional constraints that restrict the coupled decision variables by $x_{ij}^{n} \le \overline{x}_{ij}^{n}$.

In this manner, the multipliers are updated using only the information from other subsystems.

The objective difference between **SUB$_i$** and the cooperative solution implies the compromise gap between subsystem *i* and other subsystems. If it is zero, subsystem *i* can reach its solution with no conflict at the current iteration. In step 1.2, if the compromise gap is larger than the value at the prevous iteration, the step length is reduced using the step parameter. Then, in step 1.3, the step size is calculated using the compromise gap and the total amount of the conflicts between **SUB$_i$** solution and the solutions of other subsystems on the coupling constraints. The conflicts can be quantified by $\sum_{j \in T_i} (\overline{r}_j^{\,n} - A_{ij} x_{ij}^{n^*})^2$ where $\overline{r}_j^{\,n} = \max(0, (a_j - \sum_{k \in S_j - \{i\}} \overline{y}_{kj}^{n-1}))$. The conflicts are interpreted as the locally measured amount of violations when applying the solutions of all the subsystems on the coupling constraints. For each violation on the coupling constraints, the associated Lagrangian multiplers are updated using the step size. The penalty weights are determined using the Lagrangian multipliers. Then, the information vectors in the prescribed format are sent back to the coupled subsystems. Note that, at the first iteration, every subsystem solves its problem without any information from the others and thus no penalty weights will be calculated.

The iteration continues until the algorithm satisfies the predetermined stopping criteria (step 2). In this research, three stopping criteria are applied: the algorithm stops when (i) it achieves a compromised solution, (ii) the step length becomes small enough, that is $\tau^n \le \varepsilon$ where ε is a predetermined positive real constant, or (iii) the number of the direct interactions among subsystems reaches a predetermined value, $n + 1 = N$. A compromised solution is the set of the subsystems' solutions that do not violate the coupling constraints, if entered into those constraints. That is, it is a globally feasible solution even if it may not be globally optimal.

CHAPTER III

DISTRIBUTED CAPACITY ALLOCATION PROBLEM USING DIRECT INTERACTION FOR PARTIAL INFORMATION SHARING

1. Introduction

This chapter deals with a capacity allocation problem in a distributed manufacturing system environment. Specifically, multiple subsystems compete to acquire the service times of a single facility necessary for manufacturing individual products. The information associated with each product (e.g. processing times, profits, demands, etc.) is locally known only to the subsystem in charge of its production. In each subsystem, there exists a decision maker who manages local issues and makes a schedule of the production to satisfy local demands and maximize local profits. Subsystems must interact to find a compromised allocation solution that effectively utilizes the capacity of the facility. The major concern of this study is how to maximize the profits of the subsystems sharing a facility without creating conflict among the local production plans.

Due to the quasi-autonomy condition and the partial information condition of the current problem, decomposition methods (e.g. Danzig-Wolfe decomposition, Lagrangian Relaxation, etc.) are not applicable. This is because they require a centralized coordination of subproblem solutions using system information without restrictions. For example, in Lagrangian Relaxation with Subgradient Optimization method, the master problem calculates a globally feasible solution as an upperbound or lowerbound using the subproblem solutions and detailed information about the coupling constraints. This is not possible in the distributed system environment under this study.

Cooperative Interaction via Coupling Agent (CICA) (Jeong and Leon, 2002) was applied to solve the distributed capacity allocation problems with partial informatio sharing. In CICA, two main entities exist: Coupled Autonomous Organization (CAO) and Coupling Agent (CA). System problems are divided into CAOs' local decision-making subproblems and CA(s)' coordinating problem. Each CAO solves its

subproblems independently and then send the partial local information to CA(s). CA has the coupling constraints associated with multiple CAOs and takes the role of coordinating subsystem solutions in global preference using only the partial system information obtained from CAOs through the interactions. The iterative solution procedure continues until the predetermined stopping criteria are satisfid. The experimental results showed that CICA works well for the corresponding problems.

As proposed in the previous chapter, Direct Interaction for Partial Information Sharing (DIPIS) is a distributed solution method designed to work under partial information sharing. Neither centralized coordination nor third party intervention is required in the solution procedure. Additionally, direct interactions among subsystems are established to share partial subsystem information.

The remainder of the chapter is organized as follows: in section 2 the notations for mathematical representation of the problem are introduced and important constraints are explained. In section 3, the solution approach is described. Experimental results follow in section 4 along with corresponding tables and explanations. The conclusion is given in section 5.

2. A distributed capacity allocation problem

This section describes the distributed capacity allocation problem and its associated formulation.

The *distributed capacity allocation* problem is the problem of allocating the fixed capacity of a facility to satisfy the demands of multiple independent subsystems. Each subsystem is managed by a different decision maker and has unique system information. To achieve close-to-optimal allocation, subsystems must cooperate with each other by sharing partial local information. In this situation, no single decision maker exists with the complete system information, thus traditional centralized optimization methods are not applicable.

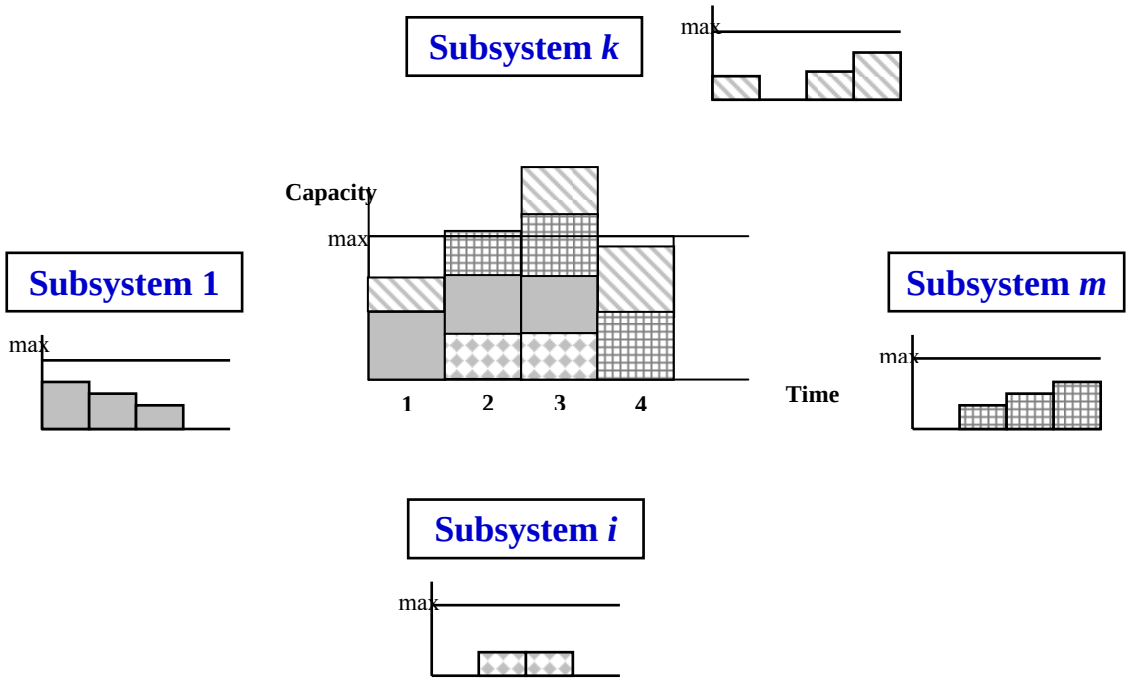

Figure 2 An example of the distributed capacity allocation problem with single facility.

Figure 2 illustrates the capacity allocation problem in a distributed system. As shown in the figure, each local production plan does not exceed the maximum capacity each time interval while the combined allocation can be infeasible due to excessive demands.

Since each subsystem must determine how many it manufactures the product to satisfy the local demands through the production horizon, the decision variable x_{it} denotes as follows:

x_{it} = production quantity of product i at time interval t proposed by subsystem i.

Additionally, the following notations are used for the mathematical representation of the associated problem:

T = total production horizon.

m = total number of subsystems in the global system.

d_i = total demand of product i of subsystem i; $i = 1,\ldots,m$.

b_{it} = profit of selling unit of product i for subsystem i at time interval t; $t = 1,..,T$.

c_t = available service time of the facility at time interval *t*.

a_{it} = processing time of unit of product *i* for subsystem *i* at time interval *t*; *t* = 1,..,*T*.

Let the global system be defined as an organization composed of associated subsystems. The objective of the global system is to maximize the sum of the subsystems' profits, $\sum_{i=1}^{m}\sum_{t=1}^{T} b_{it}x_{it}$. Two types of constraints are considered. Demand constraints confirm that each subsystem acquires sufficient production time slots in the facility to satisfy the demands within the planning horizon, $\sum_{t=1}^{T} x_{it} = d_i$, i = 1,…,*m*. Capacity constraints ensure that the total production time allocated to subsystems does not exceed the available capacity of the facility at any time period *t*, $\sum_{i=1}^{m} a_{it}y_{it} \le c_t$, *t* = 1,..,*T*. Figure 3 shows the mathematical formulation of the distributed capacity allocation problem. Note that each demand constraint is represented by the decision variables of only one subsystem, which can be categorized as a local constraint. On the other hand, the capacity constraints associated with the facility are the coupling constraints because the decision variables of multiple subsystems are involved. Under the distributed system environment herein, the most important issue is how to find close-to-optimal allocations when each subsystem has a limited view of the system. The aforementioned characteristics make the current problem more challenging. The assumptions specific to this capacity allocation problem are as follows:

- Each subsystem has direct access to only its demand information; i.e. the demand constraints, $\sum_{t=1}^{T} x_{it} = d_i$, are known only by subsystem *i* and not to subsystem *k*, $k \neq i$.
- Each subsystem has a local objective unknown to other subsystems; i.e. the objective of subsystem *i*, $\sum_{t=1}^{T} b_{it}x_{it}$ is unknown to subsystem *k*, $k \neq i$.

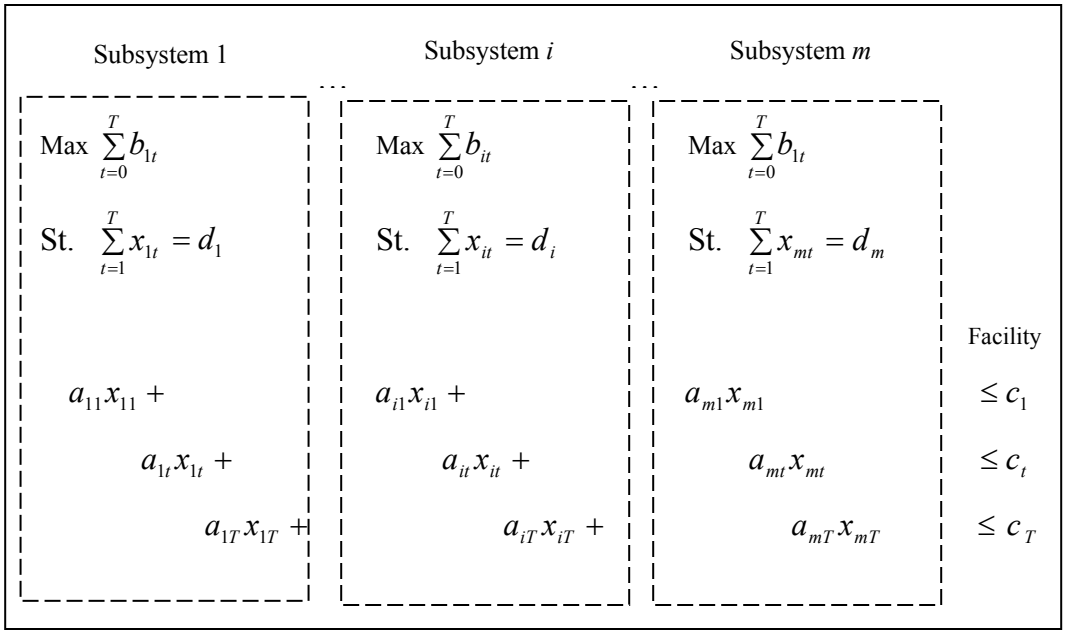

Figure 3 Mathematical representation of the distributed capacity allocation problem.

- Each subsystem has limited access to the capacity information of the facility; i.e. subsystem i can see the capacity amount of the facility for each time period, but cannot see the load of subsystem k, $k \neq i$. Each subsystem can control its load in order to not exceed the maximum available capacity in each time period; but since there is no central coordination it is possible that the total loads claimed by multiple subsystems in each time period exceed the maximum capacity of the facility.

3. Solution approach

This section describes Direct Interaction for Partial Information Sharing (DIPIS) methodology, originally proposed in Chapter II, to define the subsystem problems and solution procedures.

Under the current distributed system environment, no subsystem knows the complete machine capacity constraints of the facility. Instead, they have the partial information of decision variables and maximum available service times. DIPIS establishes direct interactions among subsystems that desire to use a facility. Through direct interactions, subsystems exchange limited local information. The information vector from subsystem

k consists of two components, the service time necessary for the required production quantity in each time interval, y_{kt}^{n-1}, and the penalty weight for not obtaining sufficient service time for the production, α_{kt}^{n-1}. Given the partial information of subsystem k, $(y_{kt}^{n-1}, \alpha_{kt}^{n-1})$, subsystem i maximizes its local objective while minimizing capacity violations by the service time requests in each interval.

Section 3.1 defines the subsystem problem in a mathematical programming formulation. In section 3.2, the procedure to calculate the penalty weights exchanged among subsystems during direct interactions is explained in detail.

3.1. *Subsystem problem* (SP$_i$)

In this section, the mathematical formulation of the subsystem problems (**SP$_i$**) is obtained by applying DIPIS method.

Let U be the index-set of the subsystems that share the facility. Then, given the information vectors $(y_{kt}^{n-1}, \alpha_{kt}^{n-1})$ from subsystem k, the problem of subsystem i can be written as,

$$\textbf{(SP}_i\textbf{)}: \text{Max} \sum_{t=1}^{T} b_{it} x_{it}^{n} + \sum_{t=1}^{T} \left(\left(\sum_{k \in U-\{i\}} \alpha_{kt}^{n-1} \right)(a_{it} x_{it} - r_{ct}) \right) \quad \text{St.} \sum_{t=1}^{T} x_{it}^{n} = d_i \tag{3.1}$$

$$\text{where } r_{ct} = \max\left(0, \left(c_t - \sum_{k \in U-\{i\}} y_{kt}^{n-1}\right)\right)$$

If complete information about the capacity constraints of the facility is available, i.e. all local subsystem information is accessible, the objective of subsystem i can be formulated as follows (Jeong and Leon, 2003):

$$\sum_{t=1}^{T} b_{it} x_{it}^{n} + \sum_{t=1}^{T} \theta_t \left(c_t - \sum_{i=1}^{m} a_{it} x_{it}\right) \tag{3.2}$$

The capacity constraint at time interval t is relaxed with Lagrangian multiplier θ_t and incorporated as an additional objective term. However, in the distributed system environment of this study, the local information of each subsystem is considered private and therefore the calculation of θ_t is not possible. Only partial information among

subsystems is shared through direct interactions. The information vector is in the following form:

$$(y_{kt}^{n-1}, \alpha_{kt}^{n-1}) \text{ where } y_{kt}^{n-1} = a_{kt} x_{kt}^{n-1} \tag{3.3}$$

Note that subsystem does not expose the full production information associated with the capacity constraints such as a_{kt} . Instead of sending a_{kt} and x_{kt}^{n-1} separately, subsystem *k* sends the combined values y_{kt}^{n-1} . α_{kt}^{n-1} reflects the marginal loss of the local objective of subsystem *k* if the desired amount of the capacity y_{kt}^{n-1} cannot be obtained. Then, using derivation similar to that which was introduced in Chapter II, the second term of (3.2) is approximated to $\sum_{t=1}^{T}\left((\sum_{k \in U-\{i\}} \alpha_{kt}^{n-1})(a_{it} x_{it} - r_{ct}) \right)$ in the problem of subsystem *i* **(SP$_i$)**.

3.2. *Penalty weights and Lagrangian multipliers update for* SP$_i$

This section explains how to update the Lagrangian multipliers and the penalty weights using only partial information from other subsystems.

Let x_{it}^{n*} be the solution of (3.1) at n^{th} iteration. During the interaction, $y_{it}^{n*} = a_{it} x_{it}^{n*}$ is sent to other subsystems at the end of the iteration with the associated penalty weight α_{it}^{n}. Note that α_{it}^{n} has occurred not from the deviation from x_{it}^{n*}, but rather from y_{it}^{n*}. Hence, given y_{kt}^{n-1} from subsystem *k* where $k \in U - \{i\}$, the second term of (3.2) is restated for subsystem *i* at n^{th} iteration as the following:

$$\sum_{t=1}^{T} \theta_t (c_t - \sum_{i=1}^{m} a_{it} x_{iit}^{n*}) \approx \sum_{t=1}^{T} \theta_{it}^{n} (c_t - \sum_{k \in U-\{i\}} y_{kt}^{n-1} - y_{it}^{n*}) \text{ where } y_{_{it}it}^{n*} = a_{it} x_{it}^{n*} \tag{3.4}$$

Note that θ_{it}^{n} is the Largrangian multiplier updated by subsystem *i* at n^{th} iteration. Then, as the weight to represent the cost increment of $\theta_{it}^{n}(c_t - \sum_{k \in U-\{i\}} y_{kt}^{n-1} - y_{it}^{n})$ when y_{it}^{n} is decreased Δ from y_{it}^{n*}, α_{it}^{n} can be derived:

$$\alpha_{it}^{n} = \left. \frac{-\partial(\theta_{it}^{n}(c_t - \sum_{k \in U-\{i\}} y_{kt}^{n-1} - y_{it}^{n}))}{\partial y_{it}^{n}} \right|_{y_{it}=y_{it}^{n*}-\Delta} = \theta_{it}^{n} \tag{3.5}$$

For the capacity violation on t^{th} time interval, θ_{it}^{n} is updated as follows:

$$\theta_{it}^{n} = \max(0, \theta_{it}^{n-1} - s^{n}(r_{ct}^{n} - y_{it}^{n*})) \tag{3.6}$$

$$\text{where } r_{ct}^{n} = \max(0, (c_t - \sum_{k \in U-\{i\}} y_{kt}^{n-1}))$$

$$s^{n} = \frac{\tau^{n} \left| Z_i^{n} - Z_{LB}^{n} \right|}{\sum_{t=1}^{T} (r_{ct}^{n} - y_{it}^{n*})^2} \tag{3.7}$$

$$\tau^{n} = \tau^{n-1} \text{ if } R_i^{n} \le R_i^{n-1},$$

$$= \tau^{n-1} \frac{R_i^{n-1}}{R_i^{n}} \text{ otherwise,} \tag{3.8}$$

$$\text{where } R_i^{n} = \frac{\left| Z_i^{n} - Z_{LB}^{n} \right|}{\sum_{t=1}^{T} (r_{ct}^{n} - y_{it}^{n*})^2} \text{ and } R_i^{n-1} = \frac{\left| Z_i^{n-1} - Z_{LB}^{n-1} \right|}{\sum_{t=1}^{T} (r_{ct}^{n-1} - y_{it}^{n-1*})^2}$$

Note that the formulas are derived from the traditional subgradient optimization method modified to work under the partial information condition. In (3.6), r_{ct}^{n} is the remaining capacity available to subsystem *i* on time interval *t* at n^{th} iteration. (3.7) is revised to update step size using local objective values. Specifically, Z_i^{n} is the objective value of subsystem *i* and Z_{LB}^{n} is the objective value of subsystem *i* obtained when it has allocated the facility capacity amount to other subsystems as requested and then used the remaining capacity for its production. If $\left| Z_i^{n} - Z_{LB}^{n} \right| = 0$, it means that subsystem *i* can maximize its objective, i.e. there is no conflict with other subsystems when using the facility. Also, the step length is reduced by the step parameter whenever $\left| Z_i^{n} - Z_{LB}^{n} \right|$ has failed to improve compared to the previous interaction, as shown in (3.8). The procedure

stops if the step length becomes small enough; that is $\tau^n \le \varepsilon$ where ε is a predetermined positive real constant. In this manner, Lagrangian multipliers can be successfully updated without the global information distributed systems manifestly lack.

The formulas (3.8) are based on the updating rule proposed by Choi and Kim (1999). Because Lagrangian Relaxation method does not guarantee the primal feasibility of the solution, we apply Sherali and Choi's (1996) procedure to recover it. Let $\boldsymbol{X}_i^n$ be the primal solution proposed by subsystem *i* at n^{th} interaction. Then, the primal solutions are defined as follows:

$$X_i^n = \sum_{k=1}^{n} \frac{x_i^k}{n} \quad \forall i \qquad i = 1,..., m \tag{3.9}$$

$\boldsymbol{x}_i^j$ is the solution of (3.1) at k^{th} iteration $j = 1,\ldots,n$. (3.9) implies that the primal solution is recovered by giving equal weight to subsystem *i*'s solutions as obtained from the first to n^{th} iteration. For more details about the primal recovery of Lagrangian solutions and the multiplier update rule, refer to Choi and Kim (1999), Sherali and Choi (1996), and Jeong (2001).

3.3. *Steps of DIPIS for distributed capacity allocation problems*

This section summarizes the proposed DIPIS algorithm for distributed capacity allocation problems.

Initialization: Set the number of maximum iterations *N*.

Set $\alpha_{it}^0 = 0, \forall i, t$, $s^0 = 0$, $\tau^0 = 2$ and p, $0 < p < 1$.

Set $n = 1$.

Step 1 Subsystem's problem (SP$_i$). For all $i = 1,..,m$

Step 1.1 Solve **SP$_i$** as shown in (3.1).

Step 1.2 Update step length ψ_i^{n-1} as shown in (3.8).

Step 1.3 Update step size t_i^{n-1} as shown in (3.7).

Step 1.4 Update ρ_i^n as shown in (3.6).

Step 1.5 Calculate α_{it}^n as shown in (3.5)

Step 1.6 Distribute y_{it}^n and $\alpha_{it}^n \ \forall i,t$ to subsystem $k \ \forall k$.

Step 2. If any of the specified stopping criteria is satisfied, stop. Otherwise $n = n + 1$ and go to step 1.

The solution procedure starts with the initialization: setting all the penalty weights equal to zero and other parameters to starting values. At every iteration, each subsystem solves its local problem and updates the penalty weights, as shown in step1. For the penalty weight updates, the subsystems need to reach a cooperative solution which minimizes the conflict among them. The cooperative solution of subsystem i at n^{th} iteration can be calculated as follows:

(i) Substitute the solutions of other subsystms received at the end of the previous iteration, $y_{kt}^{n-1}, \forall k \in u - \{i\}, t$, into the coupling constraints.

(ii) Then, solve the local problem by applying additional constraints that restrict the coupled decision variables by $a_{it} x_{ij}^n \le r_{ct}^n$ where $r_{ct}^n = \max(0, (c_t - \sum_{k \in U - \{i\}} y_{kt}^{n-1}))$.

(iii) The resulted objective value is used as the lower bound, Z_{LB}^n when the stepsize and step length are updated.

To calculate step size, the conflicts among subsystems must be quantified as $\sum_{t=1}^{T} (r_{ct}^n - y_{it}^{n*})^2 = \sum_{t=1}^{T} (r_{ct}^n - a_{it} x_{it}^{n*})^2$ where $r_{ct}^n = \max(0, (c_t - \sum_{k \in U - \{i\}} y_{kt}^{n-1}))$. The conflicts are interpreted as the locally measured amount of capacity violations when applying the solutions of all the subsystems on the capacity constraints. For each violation on the coupling constraints, the associated Lagrangian multiplers are updated using the step size. The penalty weights are determined using the Lagrangian multipliers. Then, the information vectors in the prescribed format are exchanged. Note that, at first iteration, every subsystem solves its problem without outside information thus no penalty weights will be calculated at the first iteration.

The iteration continues until the algorithm satisfies the predetermined stopping criteria (step 2). In this research, three stopping criteria are applied: the algorithm stops when (i) it achieves a compromised solution, (ii) the step length becomes small enough, that is, $\tau^n \leq \varepsilon$ where ε is a predetermined positive real constant, or (iii) the number of the direct interactions among subsystems reaches a predetermined value, $n + 1 = N$. A compromised solution is the set of the subsystems' solutions without any conflict regarding the capacity of the facility; meaning, a solution is reached when there are no violations on the capacity constraints, if the solution were applied to them. Thus, it is a globally feasible solution even if it may not be globally optimal.

4. Experimental results

This section covers the experimental study used to investigate the performance of DIPIS when solving distributed capacity allocation problems.

The proposed algorithm is tested on randomly generated problems using multiple varying factors to examine what, if any, variations affect the performance of the algorithm. The results are compared with the those of Lagrangian Relaxation (LR) algorithm and CICA. The comparison with LR results is interesting for two reasons: (i) the LR-based algorithm can use the system information without restrictions, (ii) the master problem in the LR algorithm coordinates the subsystem solutions. The main difference between DIPIS and CICA is the existence of third party mediation in the algorithm. We expect to see the effect of that in the results.

4.1. *Performance measures*

Since DIPIS does not guarantee the global feasibility and the optimal convergence, two performance measures are introduced to evaluate the quality of the solution: Percent Deviation (PD) and Capacity Violation (CV) (Jeong and Leon, 2003).

PD indicates the closeness of the solutions to the optimal solution or best solution if the optimal solution is not available.

$$\text{PD} = \frac{\left|Z - Z^*\right|}{Z^*} \times 100(\%) \tag{3.10}$$

Here Z and Z^* are the global objective value of DIPIS solution and the optimal solution, respectively.

CV is proposed to show the degree of the capacity violation of the subsystems' solutions as follows:

$$\text{CV} = \frac{\sum_{t=1}^{T} \max\left(0, \left[\sum_{i=1}^{m} a_{it} x_{it}\right] - c_t\right)}{\sum_{t=1}^{T} c_t} \tag{3.11}$$

CV represents the ratio of the sum of excessive capacity to the total facility capacity.

Each measure can reveal only one aspect of the solution quality. For example, a solution with a minimal PD but large CV may not be considered "good". Therefore, the quality of solutions must be evaluated using both measures.

4.2. *Experiments*

In this experimental study, distributed capacity allocation problems with two subsystems in a single facility are considered. DIPIS algorithm stops when n = 100 or step length $\tau^n \le \varepsilon$ where $\varepsilon = 0.00001$.

The test problems are randomly generated by varying five factors: the number of time intervals, profit of the products, processing times, demands, and capacity ratio per time interval. Two levels of variances are considered per factor as follows (Jeong and Leon, 2003):

(i) The number of time intervals is 5 or 10.

(ii) The profits are randomly generated from the uniform distribution U(10,30) or U(10,50).

(iii) The processing times are randomly generated from the uniform distribution U(2,5) or U(2,10).

(iv) The demands are randomly generated from the uniform distribution U(300, 500) or U(300,600).

(v) The capacity ratio (E) is 1.0 or 1.1.

The capacity ratio represents the ratio of maximum available facility capacity to the total demands of all subsystems. Setting the maximum available capacity to be greater than or equal to total demands, meaning $E \geq 1.0$, ensures that every problem considered herein has at least one feasible solution. Therefore, 2^5 problem types are considered and 10 problem instances are generated per problem type, totaling 320 test problems. C language is used to code the programs, and the subsystem problems in the LP formulation are solved using Cplex.

Tables 1 and 2 show the PDs and CVs of DIPIS with the results of other solution algorithms such as LR and CICA. Here the results of CICA are obtained from CICA-WPCI setting which implies that the coupled autonomous organizations (CAOs) recognize the maximum available capacity per time interval. For a fair comparison with CICA, it would be appropriate to consider the solutions of both coupled autonomous organizations and coupling agents (CA). This is because in DIPIS no third party coordinating entity such as a coupling agent exists; instead, subsystems perform the roles of both. The average PD and CV of the proposed algorithm are less than 2% and 3%, respectively. As shown in Table 1, DIPIS results are better than LR, which is unexpected. Note that LR is a centralized algorithm which can use more system information and update Lagrangian multipliers with a globally feasible lowerbound. It is conjectured that the number of subsystems is two, which is not sufficient to show the difference between the distributed system model and the centralized system model. That is, the distributedness of system information and decision entities appears to be too low. Additionally, DIPIS finds 34 optimal solutions out of the 320 problems.

5. Conclusion

In this chapter, the capacity allocation problem in a distributed environment is studied for the adaptability of the proposed distributed solution method in linear programming

formulation. To resolve the potential oscillation problem due to the LP formulation, a convex combination rule proposed by Choi and Kim (1999) is revised to update the Lagrangian multipliers in the DIPIS method.

The experimental results show that DIPIS works well for the 320 test problems. It even outperforms the centralized algorithm LR which has more system information at any given time. The conjectured reason is that the test system consists of only two subsystems, the simplest distributed system setting. Additionally, the designs of a Lagrangian multiplier update routine (which maximize the usefulness of global coordination by the master problem) are expected to improve the performance of the LR algorithm.

The capacity allocation problems in distributed systems with more than two subsystems associated with multiple facilities are suggested as a direction for future research. The results are believed to more precisely describe the performance of the proposed algorithm.

Table 1 Comparison of PD results of DIPIS with LR and CICA

	PD (%)			
	DIPIS	LR	CICA – ORG	CICA - CA
Min	0.00	0.00	0.00	0.06
Avg	1.76	4.08	2.61	8.70
Max	23.05	42.88	25.06	60.05

Table 2 Comparison of CV results of DIPIS with LR and CICA

	CV (%)		
	DIPIS	LR	CICA-ORG
Min	0.00	0.00	0.00
Avg	2.12	8.33	7.62
Max	31.71	42.72	45.38

CHAPTER IV

SINGLE-MACHINE DISTRIBUTED SCHEDULING USING DIRECT INTERACTION FOR PARTIAL INFORMATION SHARING

1. Introduction

This chapter addresses a single-machine scheduling problem that arises in a manufacturing system where the decision authorities and information are distributed in multiple sub-production systems. The situation occurs in practice when multiple sub-production systems must share a single-machine due to the limited production resources. Although sub-production systems have distinct products for independent manufacturing, some jobs require products be processed on a shared machine. Consequently, to find a compromised schedule for using the shared machine, quasi-autonomous decision makers in sub-production systems must consider interactions one another.

Direct Interaction for Partial Information Sharing (DIPIS), as proposed in Chapter II, is applied to solve this problem. Direct interactions are established among sub-production systems for sharing local information. During the interactions, only local information limited to the jobs processed on the shared machine is exchanged. The information must represent the needs of the sub-production system and supportive evidence based on the local objective and constraints. Hence, the goal of optimizing the schedule of the shared machine, as well as local sub-production objectives, is pursued.

The chapter is organized as follows. In section 2, the definition of a single-machine distributed scheduling problem is introduced in a mathematical form with necessary notations. Major issues of the problem are also described. A solution method using DIPIS is represented in section 3. The experimental results and conclusion are summarized in section 4.

2. A single-machine distributed scheduling problem

A single-machine distributed system consists of *m* sub-production systems that share a machine in processing jobs. The problem is how to schedule sub-production jobs on the shared machine in order to minimize a global objective in a linear function of job completion times. Two characteristics denote this problem as distributed: (1) multiple quasi-autonomous decision entities are involved in the problem solving, (2) there is only partial information sharing allowed. The first condition means each sub-production system has a decision maker that only schedules jobs within that sub-production system. The second condition means that the job information of each sub-production system (e.g. processing times, precedence relationships, etc.) is private and only non-private local information is partially shared with other systems in pursuit of the global goal. For example, in Figure 4, the local information about the jobs in the shaded box is shared each other.

For clarity of the presentation and derivations, total weighted completion time is examined as the objective function. Next, the notation is introduced for the mathematical representations of the problem. We use the notation similar to the one proposed by Pritsker *et al.* (1969) for IP formulation. Let *T* be the production planning horizon and the decision variable is,

$$x_{jt} = \left.\begin{cases} 1, \text{ if job } j \text{ has started by time } t. \\ 0, \text{ otherwise.} \end{cases}\right\} \; t = 1,..,T$$

The following notations are additionally used for the problem formulation:

$s_j = \sum_{t=1}^{T} (1 - x_{jt})$: starting time of job j.

U: the set of all jobs that must process on the shared machine.

U_i: the set of jobs of sub-production system i. $U_i \subset U$, $i = 1,...,m$

$n_i = |U_i|$

$N = |U| = \sum_{i=1}^{m} n_i$

p_j: processing time of job j.

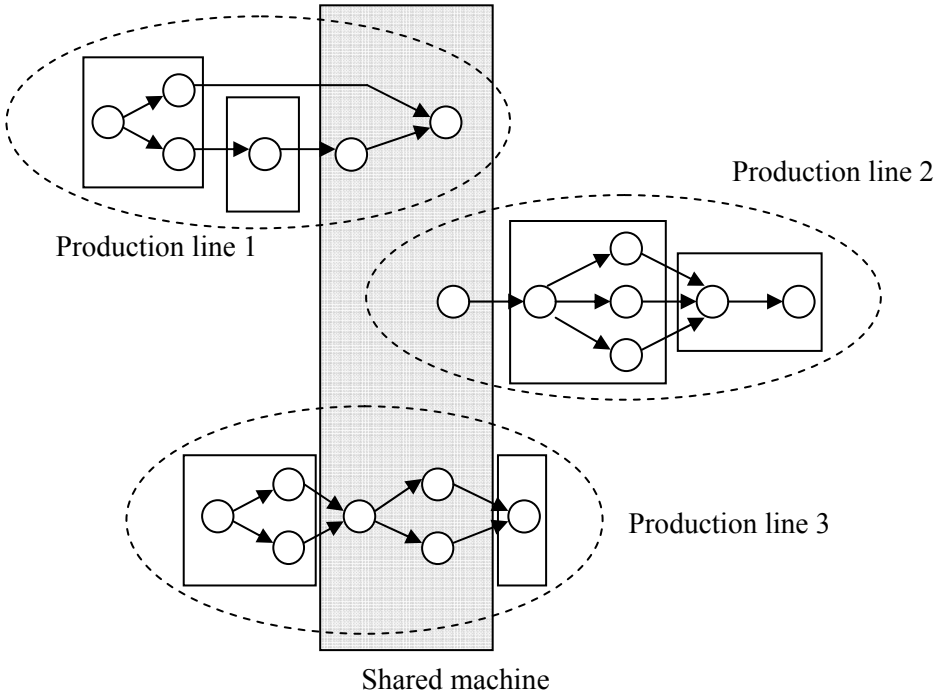

Figure 4 Single machine scheduling in a system of 3 production lines.

$c_j = s_j + p_j = \sum_{t=1}^{T}(1 - x_{jt}) + p_j$: completion time of job j.

(j, k): job j must precede job k.

w_j: weight for unit completion time of job j.

Then, the mathematical model of the single-machine scheduling problem under study is,

$$\text{Min} \quad \sum_{j \in U} w_j \left(\sum_{t=1}^{T}(1 - x_{jt}) + p_j \right) \tag{4.1}$$

$$\text{St.} \quad x_{jt+1} \geq x_{jt} \quad \forall j, t = 1,\ldots,T-1 \tag{4.2}$$

$$\sum_{t=1}^{T-p_j+1} x_{jt} \geq 1 \qquad j = 1,\ldots,N \tag{4.3}$$

$$\sum_{t=1}^{T}\left(x_{qt}-x_{rt}\right)\geq p_q \quad \forall(q,r): q,r\in U_i, i=1,...,m \tag{4.4}$$

$$\sum_{j=1}^{N}(x_{jt}-x_{jt-p_j})\leq 1 \quad \forall t \tag{4.5}$$

Quasi-autonomy condition (4.6)

Partial Information Sharing (4.7)

$$x_{jt}=\{0,1\} \qquad \forall j,\forall t \tag{4.8}$$

The objective of the system (4.1) is to minimize the total sum of the weighted completion times of all jobs processed on the shared machine. Constraint (4.2) implies that, once started, a job remains started in all subsequent time periods. Constraint (4.3) ensures that all the jobs can be finished within the production planning horizon. The precedence constraint (4.4) implies that the interval of the starting times between job q and job r must be at least p_q in order to finish job q before job r starts processing. Note that only the jobs in the same sub-production system are related by the precedence constraints, thus grouping them per each sub-production system. Constraint (4.5) is the capacity constraints of the shared machine, meaning that only one machine is available for job processing through all the processing time slots. Multiple sub-production system decision variables are included in these constraints. That is, multiple sub-production systems are coupled by the capacity constraints. (4.6) and (4.7) are the conditions that define the distributed system. Finally, Constraints (4.8) is for the declaration of the binary integer decision variables.

In the distributed scheduling problem, jobs processed on the shared machine are distributed among sub-production systems. So the local objective of sub-production system i is to minimize the sum of the weighted completion time of its own jobs on the shared machine, i.e. $\sum_{j\in U_i} w_j\left(\sum_{t=1}^{T}(1-x_{jt})+p_j\right)$. Sub-production system i recognizes only the precedence constraints between its own jobs, $\sum_{t=1}^{T}\left(x_{qt}-x_{rt}\right)\geq p_q \quad \forall(q,r): q,r\in U_i$.

These local precedence constraints are unknown to other sub-production systems. On the other hand, capacity constraints (4.5) include the decision variables of multiple sub-production systems. Each sub-production system can see its own decision variables and the available capacity per time slot, i.e. $\sum_{j \in U_i} (x_{jt} - x_{jt-p_j}) \le 1 \quad \forall t$. With these partial capacity constraints, the sub-production system problem can be solved independently. But the resulting schedule may have conflicts because multiple sub-production systems can claim the same time slots. The key issue is how to optimize sub-production systems' local schedules without creating capacity violations. Consequently, the research problem in this chapter is how to allocate jobs to time slots on a shared machine in order to minimize the total weighted completion times using partial information sharing.

3. A Solution methodology using DIPIS

This section describes the solution procedures for applying DIPIS to the single-machine distributed scheduling problem.

DIPIS relaxes the machine capacity constraints from the system problem and thus decomposes it into *m* sub-production system problems. The sub-production system problems do not have the machine capacity constraints. Instead, they include penalty functions to reflect the machine capacity information as additional objectives. For this purpose, the sub-production systems interact by exchanging local information vectors. Figure 5 illustrates direct interactions among sub-production systems and the information flows at n^{th} iteration. Let S_t be the index-set of sub-production systems that are coupled in the machine capacity constraint on time slot *t*. Sub-production system *i* receives information vectors from sub-production system *k* where $k \in S_t - \{i\}$ via direct interactions. Let Γ_k^{n-1} be the set of time slots on the shared machine that sub-production system *k* desires to use for its job processing at $(n - 1)^{th}$ iteration. The information vector from sub-production system *k* consists of two components. This first is a solution vector of sub-production system *k*'s scheduling problem composed of the desired time

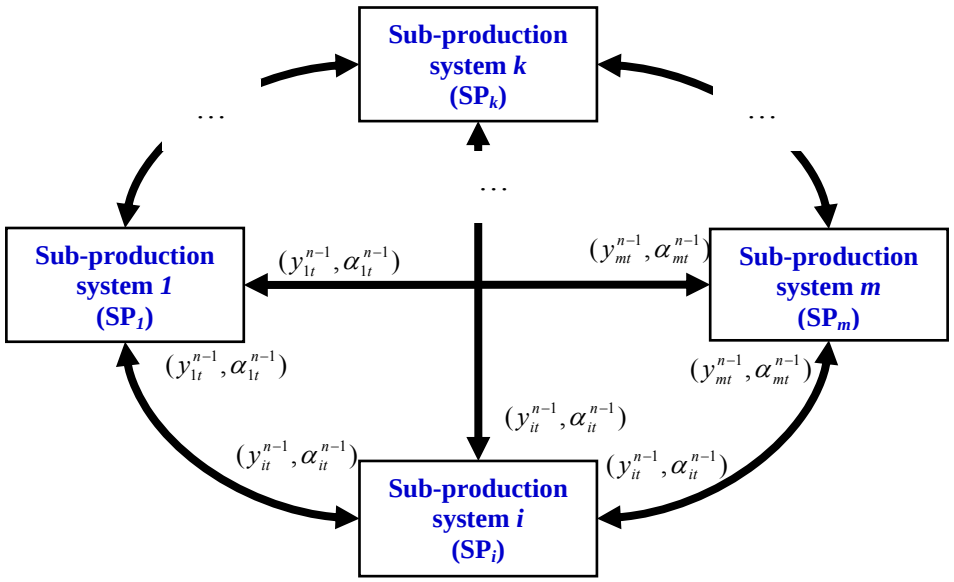

Figure 5 Information flow in DIPIS model for a single machine distributed scheduling problem with *m* sub-production systems.

slot *t*, where $t \in \Gamma_k^{n-1}$, of the shared machine for processing its jobs. The second is the weight α_{kt}^{n-1} that sub-production system *k* claims as the penalty cost of taking time slot *t*.

Section 3.1 describes the sub-production system problem in mathematical programming formulations. Section 3.2 explains the derivation of the penalty costs exchanged among sub-production systems via direct interactions.

3.1. *Sub-production system problems*

The sub-production system problem is defined in two types of mathematical programming formulations: 0/1 integer, and linear program formulation. In general, an IP problem is better suited for the scheduling problems in terms of solution quality. However, as the problem size increases the computation times increases exponentially. The proposed LP heuristic is designed to solve the problem of large sizes with practical computation times.

3.1.1 *IP formulation*

Using the same notations as in the previous section, the problem of sub-production system i (**SP$_i$-IP**) at n^{th} iteration is formulated as follows:

$$\textbf{(SP}_i\textbf{-IP):}\quad \text{Min}\quad \sum_{j\in U_i} w_j\left(\sum_{t=1}^{T}(1-x_{jt}^{n})+p_j\right)+\sum_{k\in S_t-\{i\}}\sum_{t\in\Gamma_k^{n-1}}\left(\alpha_{kt}^{n-1}\sum_{j\in U_i}(x_{jt}^{n}-x_{jt-p_j}^{n})\right) \tag{4.9}$$

$$\text{St.}\quad x_{jt+1}\geq x_{jt} \qquad \forall j\in U_i, t=1,...,T-1 \tag{4.10}$$

$$\sum_{t=1}^{T-p_j+1} x_{jt}\geq 1 \qquad \forall j\in U_i \tag{4.11}$$

$$\sum_{t=1}^{T}\left(x_{qt}^{n}-x_{rt}^{n}\right)\geq p_q \qquad \forall (q,r): q,r\in U_i \tag{4.12}$$

$$\sum_{j\in U_i}(x_{jt}^{n}-x_{jt-p_j}^{n})\leq 1 \qquad \forall t \tag{4.13}$$

$$x_{jt}^{n}=\{0,1\} \qquad \forall j,\forall t \tag{4.14}$$

In the second term of the objective function, $\sum_{j\in U_i}(x_{jt}^{n}-x_{jt-p_j}^{n})$ indicates if sub-production system i desires to use time slot t. If penalty α_{kt}^{n-1} received from sub-production system k is positive, that means sub-production system k also wants to use the slot t. Thus, it indicates the potential schedule conflict between sub-production systems i and k on time slot t. It costs α_{kt}^{n-1} for sub-production system i to take the time slot. Therefore, the problem is to find a compromised solution that minimizes both the schedule conflict with other sub-production systems and the local objective function. Constraint (4.10) ensures the nonpreemptiveness of the solution schedule. Constraint (4.11) implies that all jobs must be scheduled within the planning horizon. Constraint (4.12) and (4.13) are the local precedence constraint and the local machine conflict constraint, respectively.

Note that the second term of the objective function is the penalty function that substitutes the Lagrangian relaxed machine capacity constraint function as follows:

$$-\theta_t^n\left(1-\sum_{k=1}^{m}\left(\sum_{j\in U_k}(x_{jt}^n-x_{jt-p_j}^n)\right)\right) \tag{4.15}$$

where θ_t^n is the Lagrangian multiplier of the machine capacity constraint on the time slot t at n^{th} iteration. The Lagrangian relaxation method needs complete sub-production system information to calculate the global upperbound and update Lagrangian multiplers. But job processing times are private sub-production system information here. So, DIPIS is applied to find an alternative penalty function which can be formed using only solutions of other systems, associated sub-production system penalties, and local job information. Derivation of the penalty function using DIPIS is illustrated as follows:

$$-\theta_t^n\left(1-\sum_{k=1}^{m}\left(\sum_{j\in U_k}(x_{jt}^n-x_{jt-p_j}^n)\right)\right)=-\sum_{k\in S_t}\theta_{kt}^n\left(1-\sum_{j\in U_k}(x_{jt}^n-x_{jt-p_j}^n)\right)$$

$$\approx-\sum_{k\in S_t-\{i\}}\alpha_{kt}^{n-1}\left(r_t^n-\sum_{j\in U_i}(x_{jt}^n-x_{jt-p_j}^n)\right) \tag{4.16}$$

where $r_t^n=\max(0,1-\sum_{k\in S_t-\{i\}}(\sum_{j\in U_k}(x_{jt}^n-x_{jt-p_j}^n)))$. r_t^n indicates the remaining machine capacity on time slot t that sub-production system i can use without penalty at n^{th} iteration. By the solution vectors received via direct interactions, sub-production system i can recognize which time slots are desired by sub-production sytem k and thus determine r_t^n . Here θ_t^n is linearly approximated as $\sum_{k\in S_t}\theta_{kt}^n$ using the first order Taylor expansion and θ_{kt}^n is the partial penalty weight associated with the variations of local objective and local constraints in sub-production system k. These partial terms can be calculated by a central decision entity who does not exist in the system under study. Therefore, each term θ_{kt}^n is replace by the value, α_{kt}^{n-1}, which is locally determined by each sub-production system at the end of the previous iteration. Note that, for the subproblem of sub-production system i, its own term is set to zero and the resultant penalty weight is $\sum_{k\in S_t-\{i\}}\alpha_{kt}^n$. This is because the violations on the coupling constraints are presumed to be incurred by other sub-production systems and thus each sub-production system doesnot penalize itself.

In single-machine problems, r_t^n can be either 0 or 1. If at least one sub-production system other than sub-production system i desires to use time slot t for its job processing, $\sum_{k \in S_t - \{i\}} (\sum_{j \in U_k} (x_{jt}^n - x_{jt-p_j}^n))) \geq 1$ and r_t^n is 0. The associated penalty cost is $\sum_{k \in S_t - \{i\}} \alpha_{kt}^{n-1}$. If no other sub-production system claims time slot t, r_t^n is 1 and thus sub-production system i can use time slot t without any penalty, i.e. $\sum_{k \in S_t - \{i\}} \alpha_{kt}^{n-1} = 0$. Therefore, as shown in (4.7), the following form of the penalty function can represent both cases:

$$\sum_{k \in S_t - \{i\}} \alpha_{kt}^{n-1} \left(\sum_{j \in U_i} (x_{jt}^n - x_{jt-p_j}^n) \right) \tag{4.17}$$

The algorithm based on 0/1 IP formulation of DIPIS is expected to show promising performance dealing with single machine scheduling problems under partial information sharing. This is akin to what the CICA model showed in similar problems with allowing third party mediation (Jeong and Leon, 2005). However, IP formulation has limitations in solving problems with large job set. As the number of jobs to schedule increases, the computational efficiency deteriorates significantly, thus prohibiting practical implementation of the proposed algorithm to industry-size problems. To resolve this limitation, LP-CICA, LP version of CICA algorithm, was developed using LP relaxation proposed by Dyer and Wolsey (1990) and tested. For more details about LP-CICA, refer to appendix A. Similarly, here we introduce an efficient LP heuristic for the associated scheduling problems.

3.1.2 *LP Heuristic*

This section describes an LP relaxed formulation and a feasibility restoration routine. For LP relaxed formulation, we use notation similar to Dyer and Wolsey (1990) and Uma and Wein (1998). First, let the decision variable y_{jt} be newly defined as follows:

$$y_{jt} : \begin{cases} 0 < y_{jt} \leq 1, \text{ if job } j \text{ is being processed in the time interval } [t, t+1]. \\ 0, \text{ otherwise.} \end{cases}, t = 1, \ldots, T.$$

Then, the LP problem of sub-production system i (**SP$_i$-LP**) is proposed as follows:

(SP$_i$-LP): Min $$\sum_{j\in U_i} w_j c_j^n + \sum_{k\in S_t-\{i\}} \sum_{t\in \Gamma_k^{n-1}} (\alpha_{kt}^{n-1} Y_t^n) \quad (4.18)$$

St. $$\sum_{t=1}^{T} y_{jt}^n = p_j \qquad \forall j \in U_i \quad (4.19)$$

$$c_r^n - c_q^n \geq p_r \qquad \forall (q,r): q,r \in U_i \quad (4.20)$$

$$\sum_{j\in U_i} y_{jt}^n \leq 1 \qquad \forall t \quad (4.21)$$

$$c_j^n - \frac{1}{p_j}\sum_{t=1}^{T}(t+\frac{1}{2}) y_{jt} + \frac{p_j}{2} = 0 \qquad \forall j \in U_i \quad (4.22)$$

$$Y_t^n - \sum_{j\in U_i} y_{jt}^n = 0 \qquad \forall t \quad (4.23)$$

In the objective (4.18), the variable c_j^n is used for the completion time of job j and Y_t^n is for the occupancy of sub-production system i on the time slot t at n^{th} iteration. Constraint (4.19) ensures that job j be finished within the planning horizon. The precedence constraint is shown in (4.20): the time interval between the completion times of job q and r must be at least p_r. Constraint (4.21) implies that the maximum available machine capacity per time slot is one. Using the formula (4.22), the completion time of job j is calculated. Note that, if the schedule for job j is nonpreemptive, (4.22) calculates the exact completion time. If it is preemptive, it provides the average completion time of the schedule. (4.23) defines Y_t^n to be total sum of the processing times requested by sub-production system i on time slot t.

The main differences between the LP problem and IP problem originate from the introduction of newly defined decision variables. The decision variables can now take any fractional value in [0,1]. This implies that more than one job can be processed on the same time slot. Accordingly, forcing the schedule to be nonpreemptive becomes less meaningful. Hence, in the LP problem, nonpreemptive constraints are relaxed for further improvement of the computational efficiency.

The resultant solution of **SP$_i$-LP** can be a preemptive nonintegral schedule, making it even locally infeasible. So, a feasibility restoration routine is needed to recover the integrality and the nonpreemptiveness of the schedule. In this study, a simple post-processing routine based on the completion time information from the **SP$_i$-LP** schedule is applied as follows:

Feasibility Restoration Routine.

Step 1. Sort jobs in the ascending order of the completion time.

Step 2. If two or more jobs have the same completion time, order them by WSPT rule. If jobs are still tied, order them arbitrarily.

Step 3. Check the schedule overlap between adjacent jobs. If overlapping occurs, adjust the starting time of the succeeding job to the completion time of the preceding job.

Step 4. Check if the last job finishes processing within the planning horizon. If not, readjust the completion time of the last job so that the last job finishes by the planning horizon. If overlapping occurs due to this adjustment, reset the completion time of the preceding job by the starting time of the succeeding job.

Using the feasibility restoration routine proposed above, we can convert solutions of **SP$_i$-LP** to nonpreemptive integer schedules.

3.2. *Updating the Lagrangian multiplier for the sub-production systems*

In this section, we explain how to update the Lagrangian multipliers of machine capacity constraints and calculate the penalty weights. For convenience of the illustration, we use the notations for **SP$_i$-IP**.

Let θ_{it}^n be the Lagrangian multiplier for the machine capacity constraint on time slot t calculated by sub-production system i at the end of n^{th} iteration. θ_{it}^n is updated using the subgradient optimization method modified as follows to work under partial information:

$$\theta_{it}^n = \max(0, \theta_{it}^{n-1} - s_i^n (r_t^n - \sum_{j \in U_i} (x_{jt}^n - x_{jt-p_j}^n))) \tag{4.24}$$

$$\text{where } r_t^n = \begin{cases} 0, & \text{if } t \in \Gamma_k^{n-1} \text{ for } \exists k \in S_t \\ 1, & \text{otherwise} \end{cases}$$

$$s_i^n = \frac{\tau_i^n \left| Z_i^n - Z_c^n \right|}{\sum_{t=1}^{T} \left(r_t^n - \sum_{j \in U_i} (x_{jt}^n - x_{jt-p_j}^n) \right)^2} \tag{4.25}$$

$$\tau_i^n = \tau_i^{n-1} \times p \text{, if } \left| Z_i^n - Z_c^n \right| > \left| Z_i^{n-1} - Z_c^{n-1} \right|$$

$$= \tau_i^{n-1} \text{, otherwise,} \tag{4.26}$$

where Z_i^n is the local objective value of sub-production system *i* problem at n^{th} iteration. Z_c^n is the local objective value of sub-production system *i* when the problem is further restricted to minimize the schedule conflict with other sub-production systems. For instance, when $t \in \Gamma_k^{n-1}$, then the time slot *t* is restricted in order that sub-production system *i* cannot take the time slot for its schedule. Therefore, Z_c^n is obtained when restricting all $t \in \Gamma_k^{n-1}$ for $k \in S_t - \{i\}$.

Note that the proposed Lagrangian multipliers update rule does not require a global upper bound to update step size. As shown in (4.25), the local solutions Z_i^n and Z_c^n are used to calculate the step size. $\left| Z_i^n - Z_c^n \right|$ tells the compromise level of sub-production system *i* with other sub-production systems. If $\left| Z_i^n - Z_c^n \right| = 0$, sub-production system *i* schedules its jobs without using all the time slots desired by other sub-production systems. Otherwise, Lagrangian multipliers are updated proportional to $\left| Z_i^n - Z_c^n \right|$. As shown in (4.26), step length is reduced when $\left| Z_i^n - Z_c^n \right|$ is not improved compared to the previous iteration.

The next step is to determine the penalty weight using the updated Lagrangian multipliers. The penalty weight α_{it}^n represents the average cost increment incurred when a new schedule is built without using time slot *t*. Consider a job *j* which is scheduled from $s_j^n + 1$ to $s_j^n + p_j$. The total cost increment incurred when sub-production system *i*

cannot use this time interval is $\sum_{t=s_j^n+1}^{s_j^n+p_j} \theta_{it}^n$. Then, the average cost increment in this time interval is calculated as follows:

$$\alpha_{it}^n = \frac{\sum_{t=s_j^n+1}^{s_j^n+p_j} \theta_{it}^n}{p_j} \tag{4.27}$$

(4.27) is applied to α_{it}^n for all the time slots scheduled for the jobs of sub-production system *i*, i.e. $\forall t \in \Gamma_i^n$.

3.3. *DIPIS algorithm for single-machine distributed scheduling problems*

In this section, DIPIS algorithm for a single-machine distributed scheduling problem is summarized.

Initialization: Set the number of maximum interactions *N*.

Set $\Gamma_k^0 = \{\ \}$, $\alpha_{kt}^0 = 0$, $s_i^0 = 0$, $\tau_i^0 = 2$, $\forall k, t$ and p, $0 < p < 1$.

Set $n = 1$.

Step 1: Sub-production system's problem. For $i = 1,\ldots,m$.

Step 1.1: Solve the problem **SP*i*-IP** or **SP*i*-LP** and apply the feasibility restoreation routine to find Γ_i^n .

Step 1.2: Calculate the step length τ_i^n as shown in (4.26).

Step 1.3: Calculate the step size s_i^n as shown in (4.25).

Step 1.4: Update the Lagrangian multipliers θ_{it}^n as shown in (4.24) $\forall t$.

Step 1.5: Calculate penalty weights α_{it}^n, $\forall t \in \Gamma_i^n$ as shown in (4.27).

Step 1.6: Distribute the information vector $(t, \alpha_{it}^n), \forall t \in \Gamma_i^n$ to sub-production system $k, \forall k \in S_t - \{i\}$.

Step 2: If $n = N$ or $\Gamma_i^n = \Gamma_i^{n-1}$ $\forall i$, stop. Otherwise, $n = n + 1$ and go to step 1.

The solution procedure starts with the initialization: setting all the penalty weights equal to zero and other parameters to starting values. At every iteration, each sub-production system solves its local problem and updates the penalty weights, as shown in step1. For the penalty weight updates, the sub-production systems need to reach a cooperative solution which minimizes the schedule conflict on the shared machine one another. The cooperative solution of subsystem i at n^{th} iteration can be calculated as follows:

a. For $\forall t \in \Gamma_k^n$ from sub-production system k, set $r_t^n = 0$ which means that current sub-production system would not be able to schedule its jobs on $\forall t \in \Gamma_k^n$.

b. Then, solve the restricted problem by (i). The resulted objective value is used as Z_c^n for updating step size and step length.

To calculate step size (step 1.3), the conflict among sub-production systems is quantified as the sum of capacity violations on all the time slots in the following form: $\sum_{t=1}^{T}(r_t^n - \sum_{j \in U_i}(x_{jt}^n - x_{jt-p_j}^n))^2$. For each violation on the coupling constraints, the associated Lagrangian multiplers are updated using the step size (step 1.4). The penalty weights are determined using the Lagrangian multipliers (step 1.5). Then, the information vectors in the prescribed format are exchanged (step 1.6).

The iteration continues until the algorithm satisfies the predetermined stopping criteria (step 2). In this research, two stopping criteria are applied: the algorithm stops when (i) it achieves a converged solution, (ii) the number of the direct interactions reaches a predetermined value, i.e. $n + 1 = N$. If a converged solution does not violate any of the coupling constraints, it is a compromised solution; meaning, it is a globally feasible solution even if it may not be globally optimal.

4. Experimental study

This section explains the experimental study used to investigate the performance of DIPIS in single-machine distributed scheduling problems. Specifically, two experiments are performed to test the proposed algorithm. The first experiment tests how the variance of the problem factors affect the performance of DIPIS. The second focuses on the computational efficiency of the algorithm, especially for large problems. The results are compared to those of centralized algorithm. The algorithm is implemented using C programming language. CPLEX is used to solve the sub-production system problems in LP formulation.

In this experimental study, two performance measures are proposed to evaluate the quality of DIPIS solutions: PD (Percent Deviation) and CV (Capacity Violation). PD is calculated to examine the closeness of the solution after global feasibility restoration to the global optimal solution.

$$\mathrm{PD} = \frac{\left|Z - Z^*\right|}{Z^*} \times 100(\%) \tag{4.28}$$

Here Z and Z^* are the global objective value of DIPIS solution and the optimal solution, respectively.

CV is designed to measure the capacity violations of the solutions of sub-production systems before global feasibility restoration.

$$\mathrm{CV} = \sum_{t=1}^{T} \max\left(0, \left(\sum_{k=1}^{m} \sum_{j \in U_k} (x_{jt} - x_{jt-p_j}) - 1\right)\right) \tag{4.29}$$

CV indicates the total excess of capacity by the solutions of the sub-production systems.

Finally, the computation times of DIPIS algorithm are examined and compared to the results of centralized algorithm.to check the applicability to large size problems.

4.1. *Experiment one*

This experiment considers three sub-production systems with one shared machine problem. Each sub-production system has six jobs to be processed on the shared machine.

The main purpose of this experiment is to explore the behavior of the proposed algorithm according to the variations of the problem parameters. To do so, the combinations of processing times and precedence level of jobs are considered as follows:

(1) processing times generated from U(1,5) or U(1,10)

(2) three levels of the precedence relationships

In level 1, two jobs in each sub-production system are in a precedence relationship. In level 2 and 3, three and four jobs are in the precedence relationship, respectively, The problem types are classified by the combinations of the distribution of processing time and the precedence level. For instance, '(1,5) level 1' implies that processing times are generated from U(1,5) and two jobs are in a precedence relation. Therefore, a total of 6 types of problems are examined in this experiment. The weight for completion time is randomly generated from the discrete uniform distribution U(1,10). For each problem type, 30 problems are generated and tested. We set a step parameter p = 0.75 that has performed well empirically in distributed scheduling problems (Jeong and Leon, 2002).

Table 3 and 4 show the PD, CV, and computational times of DIPIS with step parameter p = 0.75. The PD results in Table 3 illustrate that, on average, DIPIS yields solutions less than 5% from optimal.

A statistical test is performed to identify if the variations in processing times affects the solution quality of DIPIS. The first test hypotheses are as follows:

$$H_0: \mu_{PD,(1,5)} = \mu_{PD,(1,10)} \text{ and } H_1: \mu_{PD,(1,5)} < \mu_{PD,(1,10)}$$

Here $\mu_{PD,(1,5)}$ and $\mu_{PD,(1,10)}$ denote the average PDs with the problems of the processing times from U(1,5) and U(1,10), respectively. At significance level of 5 %, H_0 is rejected

with p-value equal to 0.0195, meaning the difference between two PDs are statistically significant.

The second hypotheses are designed for CV results as follows:

$$H_0: \mu_{CV,(1,5)} = \mu_{CV,(1,10)} \text{ and } H_1: \mu_{CV,(1,5)} < \mu_{CV,(1,10)}$$

Here $\mu_{CV,(1,5)}$ and $\mu_{CV,(1,10)}$ are the average CVs with the problems of the processing times from U(1,5) and U(1,10), respectively. At significance level of 5 %, H_0 is rejected with p-value equal to 0.0001. Test result tells that CVs are also significantly affected by the variations in processing times. Especially, according to the results shown in Table 3, average CVs are doubled when the variations in processing times are increased twice.

The purpose of the second statistical test is to examine the effect of the precedence levels on the solution quality. The first test hypotheses are as follows:

$$H_0: \mu_{PD,(1,5)level1} = \mu_{PD,(1,5)level3} \text{ and } H_1: \mu_{PD,(1,5)level1} < \mu_{PD,(1,5)level3}$$

Here $\mu_{PD,(1,5)level1}$ and $\mu_{PD,(1,5)level3}$ denote the average PDs with the problems of the processing times from U(1,5) and the precedence level 1 and 3, respectively. At significance level of 5 %, H_0 is not rejected with p-value equal to 0.7402.

The next test hypotheses are as follows:

$$H_0: \mu_{PD,(1,10)level1} = \mu_{PD,(1,10)level3} \text{ and } H_1: \mu_{PD,(1,10)level1} < \mu_{PD,(1,10)level3}$$

At significance level of 5 %, H_0 is rejected with p-value equal to 0.0332.

The two hypotheses test results show that the variations in precedence level does not change PDs significantly with low variations in processing times while it significantly affects PDs with high variations in processing times.

In Table 4, the average computational times of the centralized algorithm and DIPIS are compared. The results show that DIPIS solves all the problems in two minutes. Because the sizes of the test problems in this experiment are small, absolute comparison

between two results is considered to be less meaningful. The same comparison for large problem instances will be dealt in the next section.

Table 3 CV and PD results of DIPIS for small problem instances

Problem type	Percent Deviation (PD)			Capacity Violation (CV)		
	Min	avg	Max	Min	Avg	Max
(1,5) level 1	0.20	3.07	7.26	4	13.03	24
(1,5) level 2	0.30	2.92	7.11	4	15.03	29
(1,5) level 3	0.04	3.24	8.79	3	12.13	22
(1,10) level 1	0.50	3.21	8.00	7	27.30	51
(1,10) level 2	0.82	3.68	9.98	9	26.97	63
(1,10) level 3	0.77	4.53	9.74	9	24.77	49

Table 4 Computational times of DIPIS

Problem type	Computational time (sec.)			
	Optimal solution	DIPIS algorithm		
		Min	Avg	Max
(1,5) level 1	0.16	58.29	61.50	68.82
(1,5) level 2	0.17	59.81	61.78	70.02
(1,5) level 3	0.18	57.88	61.66	67.50
(1,10) level1	0.22	58.98	63.03	68.24
(1,10) level 2	0.25	60.99	67.50	89.82
(1,10) level 3	0.27	58.95	63.80	75.06

4.2. *Experiment two*

This section checks the practical applicability of DIPIS for solving large problem instances.

The experiments herein consider 36 and 48 jobs. The processing times are randomly generated from uniform distribution U(1,50) or U(1,100). Precedence level 2 is applied to all the test problems, meaning half of jobs are related with the precedence constraints. A total of seventy random problems are generated and tested.

Table 5 show the PD results for each problem type. The problem types are represented by the combination of the number of jobs and the maximum processing time. For example, '(36,50)' implies that the problem instances are to schedule 36 jobs with the processing times randomly generated from U(1,50). According to the results, DIPIS generates the solutions within 6% of optimal. Note that these results are obtained under partial information sharing and are thus encouraging.

As the processing times of jobs increase, computational efficiency of the centralized problems in IP formulations significantly deteriorates (refer to Table 6). For example, the centralized problems of 48 jobs with the processing times generated from U(1,100) requre more than 4 hours while DIPIS solves the same type of the problems in less than 2 hours.

Figure 6 illustrates the exponential increase in the computational times of centralized algorithm. Computational times of DIPIS also tend to increase but relatively small by increases in the problem size.

5. Conclusion

This study explores the performance of DIPIS for various types of single-machine scheduling problems. Besides the overall solution qualities of less than 6% from the optimal solutions, comparably excellent performances in computational times make DIPIS promising and competitive – especially for large problems.

Investigating various interaction mechanisms in a distributed system environment is interesting and worth studying as future research.

Table 5 PD and CV results of DIPIS for large problem instances

Problem type	Percent Deviation (PD)			Capacity Violation (CV)		
	Min	avg	Max	Min	Avg	Max
(36, 50)	1.50	4.08	7.33	132	215.1	430
(36, 100)	1.29	5.22	22.04	288	481.45	734
(48, 50)	1.36	3.33	6.41	246	345.45	520
(48, 100)*	2.27	3.69	8.38	578	695.6	822

*due to excessive runtimes, only ten problems are tested in this type.

Table 6 Comparison of average computational times

Problem type	Computational Times (sec.)					
	Optimal			DIPIS		
	Min	avg	Max	Min	Avg	Max
(36, 50)	66.39	141.68	307.19	336.27	409.50	549.17
(36, 100)	545.42	1109.7	2246.09	707.34	952.45	1222.6
(48, 50)	617.99	2039.75	4476.63	974.16	1311.01	1715.8
(48, 100)*	5903.53	17965.06	36874.34	2782.70	4311.27	5981.1

*due to excessive runtimes, only ten problems are tested in this type.

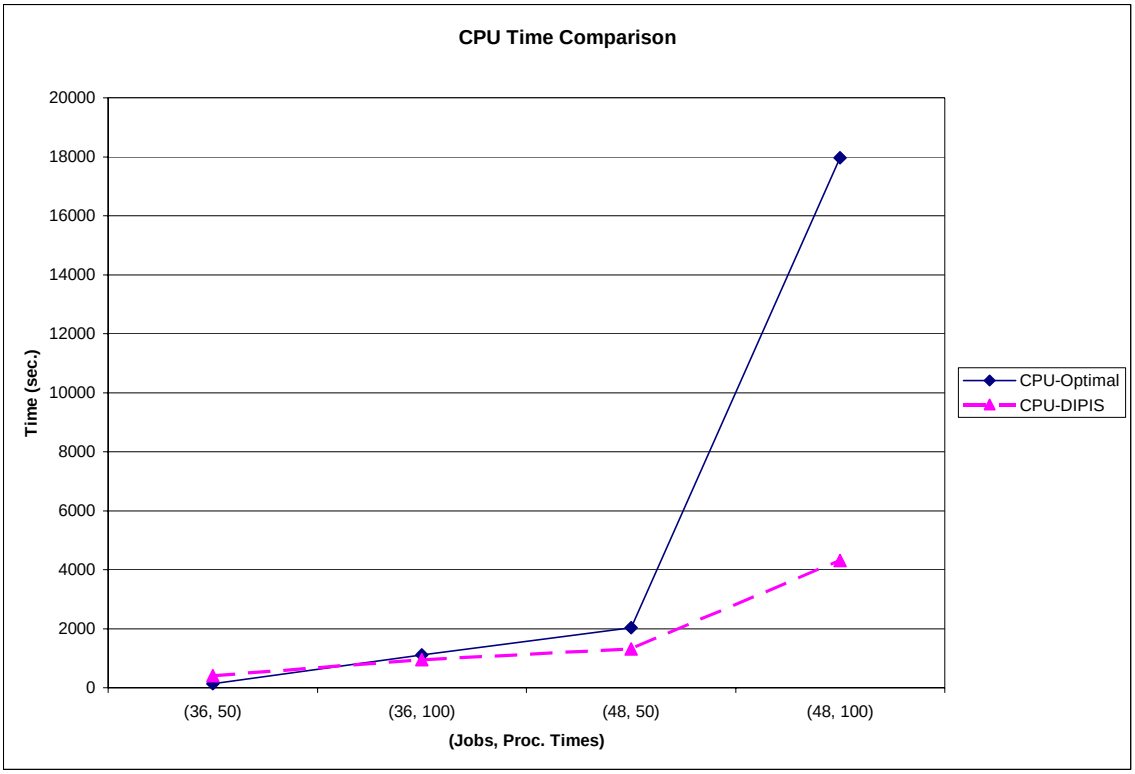

Figure 6 Computational time comparison.

CHAPTER V

DISTRIBUTED JOBSHOP SCHEDULING USING DIRECT INTERACTION FOR PARTIAL INFORMATION SHARING

1. Introduction

In jobshop manufacturing systems, multiple machines process jobs with multiple operations. The scheduling problems associated with the jobshop systems are often too complicated to be addressed by a centralized decision maker. Take the example of an assembly plant for a product consisting of multiple parts. The parts manufactured on different machine shops are assembled according to a predetermined sequence. What if some parts are produced by subcontractors ? (Ure and Jaegersberg, 2005) The machine shops of subcontractors must be managed separately by independent decision makers. According to the local machine conditions (e.g. machine capacity, downtime, etc.), they generate local production schedules. However, the parts must be delivered to the assembly plant in order that the total assembly time can be minimized. A supervisor in the assembly plant may not directly control the local information or local tasks of the subcontractors.

Lagrangian relaxation has been studied by many researchers as an efficient centralized decomposition method to jobshop scheduling problems (Roundy *et al*., 1991; Gou *et al*., 1994). But to coordinate subproblem solutions within a global preference, the master problem must be able to access local information without restriction. This is not possible in the aforementioned scenario. Cooperative Interaction via Coupling Agents, a distributed method proposed by Jeong and Leon (2002), has been applied to solve the scheduling problems in a systems where there is only partial information sharing. However, in CICA, third party entities (Coupling Agents) exist to guide subsystems in finding compromised solutions.

In this chapter, DIPIS is applied to solve the distributed jobshop scheduling problems. This method requires neither centralized coordination nor third party mediation. It uses direct interactions among subsystems to share partial information.

The chapter is organized as follows: Section 2 presents the mathematical formulation of the problem. The solution approach is described in section 3. Section 4 shows the experimental results in tables with explanations. Finally, the conclusion appears in section 5.

2. Problem statement

This section explains jobshop scheduling problems in a distributed system environment.

General jobshop scheduling problems are described as follows: N jobs are to be processed on m machines with the objective of minimizing sum of the weighted completion times of jobs, subject to two constraints: (i) the operation sequence of each job is prescribed; and (ii) each machine can process only one job at a time. (i) implies precedence constraints for job operations and (ii) indicates machine capacity constraints. An operation is denoted as the processing of a job on a machine. The completion of each job requires m operations and the operation sequence is predetermined. The operation sequences are not necessarily the same for all jobs. Each operation is non-preemptive and can only be processed on a designated machine.

In this study, we consider a distributed jobshop system that consists of m subsystems. Each subsystem possesses a unique machine to process one operation of each job. The system information each subsystem has is restricted to only the processing times of N operations on its machine and partial job operation sequences. From the partial operation sequences, subsystems can identify which subsystem processings precede or follow their job operations. The subproblem of each subsystem is to schedule the job operations it processes in order to minimize the sum of the weighted completion times subject to the machine capacity constraints. Subsystems also exchange partial local information to

achieve the global objective. As previously mentioned, this is the minimization of the sum of the weighted completion times of the last operation in each job.

The decision variables are as follows:

s_{ij} : starting time of operation i of job j

c_{ij} : completion time of operation i of job j i

C_j : completion time of job j

In addition, the following notations are used to mathematically model the problem:

T: the production planning horizon

p_{ij} : processing time of job j on machine i, $i = 1, \ldots, m$

w_j: weight for unit completion time of job j

A_j : set of pairs of operations for job j constrained by precedence relations

(k, l): operation k must precede operation l

For convenience, $T = \sum_{i=1}^{m} \sum_{j=1}^{N} p_{ij}$ in order to schedule all jobs. And the index i for an operation also indicates the subsystem that has a machine for processing that operation.

The jobshop scheduling problem under study can be represented as follows:

$$\text{Min} \quad \sum_{j=1}^{N} w_j C_j \tag{5.1}$$

$$\text{St.} \quad c_{lj} - c_{kj} \geq p_{lj} \qquad \forall (k,l) : k, l \in A_j, j = 1, \ldots, N \tag{5.2}$$

$$c_{ij} - c_{ik} \geq p_{ij} \quad \text{or } c_{ik} - c_{ij} \geq p_{ik} \qquad i = 1, \ldots, m \ , \forall j \tag{5.3}$$

$$C_j \geq c_{ij} \qquad \forall i \ , \forall j \tag{5.4}$$

Quasi-autonomy condition (5.5)

Partial Information Sharing condition (5.6)

The system objective (5.1) is to minimize the total sum of the weighted completion times of N jobs processed on m machines. The precedence constraint (5.2) implies that the interval of the starting times between operation k and l of job j must be at least p_{lj} in order to finish operation k before operation l starts processing. Note that two operations of a job in the precedence relation are processed on different machines. Since each

machine belongs to a distinct subsystem, every precedence constraint associates two subsystems and is considered to be a coupling constraint. Constraint (5.3) is the capacity constraints of the machines, meaning that a machine can process only one job operation at a time. Constraint (5.4) ensures that the completion time of a job be the time to finish processing all its operations. In a distributed problem, this constraint is not applicable because no subsystem knows completion times of all job operations. Associated with constraint (5.4), the relationship between the global objective and subsystem objectives is explained in the following section. (5.5) and (5.6) are the conditions that characterize the distributed jobshop systems: (i) each subsystem has a quasi-autonomous decisiion entity, (ii) and job processing times on a specific machine are private information only known to the subsystem that owns the machine.

In the distributed jobshop system studied here, each subsystem has multiple operations to process on its machine and generates a schedule by solving the local subproblem. Then, partial information associated with the precedence relationships among the operations of jobs is shared to optimize the global goal. Figure 7 shows the disjunctive graph representation of the corresponding problem. The first and last nodes are artificial, used for starting and ending respectively. Operations of the same color, which are connected with conjunctive arcs, are in precedence relationships. Disjunctive arcs link the operations that belong to the same subsystem for processing on one machine. Here Okj represents the k^{th} operation of job j where k = 1, 2, 3 and j = 1, 2, 3, 4.

To illustrate, let subsystem 1 be responsible for the processing the operations O11, O22, O23, O34; subsystem 2 for O21, O32, O13, O24; and subsystem 3 for O31, O12, O33, O14. Each subsystem determines the starting times of its operations without schedule conflict on its own machine. However, schedule conflicts with other subsystems may arise because no subsystem has control of the starting times of the operations in other subsystems. For example, the schedule of subsystem 1 is in the order of (O23, O11, O34, O22) and subsystem 2 in (O24, O32, O21, O13). Consider a pair of operations in precedence relationships, (O13→O23). O23 is the first operation in the

current schedule of subsystem 1 and O13 the last operation of subsystem 2. Without recognizing the completion time of O13 in subsystem 2, subsystem 1 may schedule the starting time of O23 as early as possible to minimize the weighted completion time of O23. If subsystems do not share schedule information, similar conflicts may occur between all the pairs of operations in the precedence relationship, (O11→O21), (O24→O34), and (O22→O32).

Therefore, the distributed jobshop problem is how to schedule the operations of jobs in order to minimize the sum of the weighted completion times under partial information sharing. Neither centralized coordination nor third party intervention is allowed to resolve the conflicts among the schedules of the subsystems. Interactions among subsystems are required to exchange local information in order to find solutions without the precedence violation.

3. A Solution methodology using DIPIS

This section describes Direct Interaction for Partial Information Sharing (DIPIS) methodology, proposed in Chapter II, to define the subsystem problem and solution procedures.

In examinging the relationships between the global system objective and subsystems' objectives, we consider the global objective, $\sum_{j=1}^{N} w_j C_j$,and the local objective of subsystem i, $\sum_{j=1}^{N} w_j (c_{ij} - r_{ij})$. Since w_j is known to all subsystems, we focus on the relationships between completion times of a job and the operations of the job.

By definition, the completion time of a job indicates the time to finish the last operation of the job. But the completion time of a job implicitly includes all the processing information of the intermediate operations as well as the last operation. To expand the completion time of a job using this intermediate information,

$$C_j \equiv c_{[m]j} = (c_{[1]j}) + (c_{[2]j} - c_{[1]j}) + \ldots + (c_{[i]j} - c_{[i-1]j}) + \ldots + (c_{[m]j} - c_{[m-1]j}) \qquad (5.7)$$

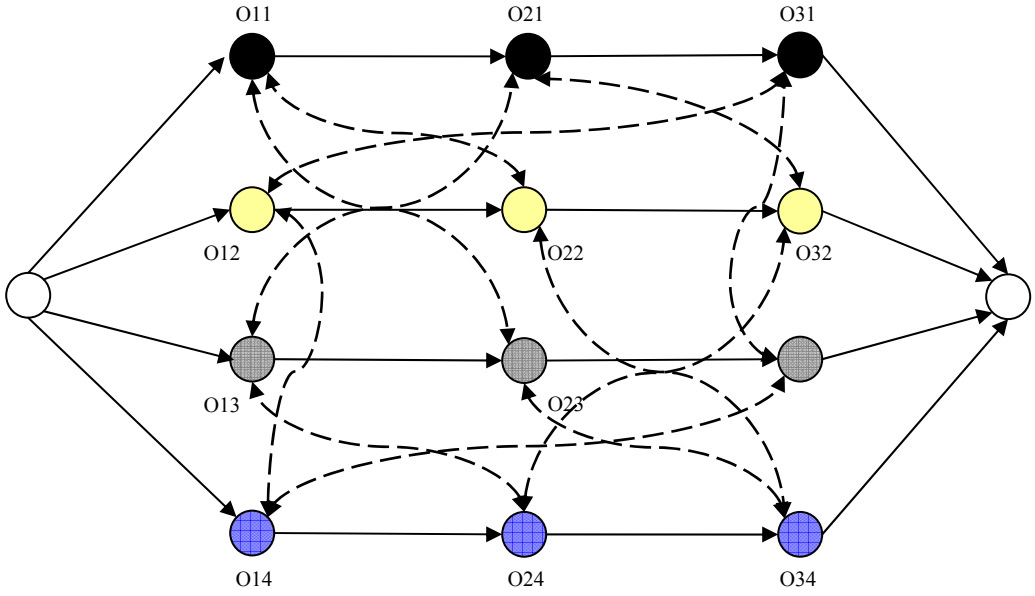

Figure 7 Disjunctive graph of the distributed jobshop problem with three subsystems.

Here $c_{[i]_j}$ means the completion time of the i^{th} operation of job j. Because the i^{th} operation of job j can only be processed after the $(i\text{-}1)^{\text{th}}$ operation of job j are finished processing, the completion time of the $(i\text{-}1)^{\text{th}}$ operation is equal to the ready time of the i^{th} operation, i.e. $r_{[i]_j} = c_{[i-1]_j}$. Note that $r_{[1]_j} = 0$ for $\forall j$. The minimization of the completion time of job j is equivalent to the minimization of the sum of the gap between the completion time of the i^{th} operation and its ready timefor i =1,…,m. Therefore, if a subsystem can obtain the ready time information of its operations, the system objective is represented as follows:

$$\min \sum_{j=1}^{N} w_j C_j = \min \sum_{j=1}^{N}\sum_{i=1}^{m} w_j (c_{[i]_j} - r_{[i]_j}) = \min \sum_{j=1}^{N}\sum_{i=1}^{m} w_j (c_{ij} - r_{ij}) = \min \sum_{i=1}^{m}\sum_{j=1}^{N} w_j (c_{ij} - r_{ij}) \tag{5.8}$$

Note that the last term means that the system objective can be represented as the sum of m subsystems' objectives.

Figure 8 shows direct interactions among subsystems and the information flows of DIPIS model for a distributed jobshop problem. Through the interactions among subsystems, partial local information associated with the precedence relations is

exchanged. Let c_{ij}^{n-1} and s_{ij}^{n-1} be the completion time and the starting time of operation *i* of job *j* on a machine of subsystem *i* after (*n*-1)th iteration, respectively. α_{ij}^{n-1} and β_{ij}^{n-1} denote the Earliness and Tardiness (E/T) weights of starting operation *i* of job *j* one time unit early and late from s_{ij}^{n-1} respectively. The information vector from subsystem *i* to subsystem *k* that has the next operation of job *j* is $(c_{ij}^{n-1}, \alpha_{ij}^{n-1})$ and the information vector to subsystem *l* that has the preceding operation of job *j* is $(s_{ij}^{n-1}, \beta_{ij}^{n-1})$. Note that c_{ij}^{n-1} is used as the proposed ready time of the next operation of job *j* and s_{ij}^{n-1} as the proposed due date of the previous operation of job *j* at n^{th} iteration

Section 3.1 describes the subsystem problem in mathematical programming formulation. In section 3.2, the procedure to calculate Earliness/Tardiness (E/T) weights is explained in detail.

3.1. *Subsystem problem* (MSP$_i$)

In this section, a mathematical formulation of the subsystem problem is introduced. Because each subsystem has one machine to schedule the operations, the subsystem problem is equivalent to a single-machine scheduling problem with *n* operations. We define the problem in LP formulation, similar to that of Chapter IV. The main difference between these two formulations is the relaxed constraints: the formulation in Chapter IV relaxes the machine capacity constraint and the one defined here relaxes the precedence constraints. The same feasibility restoration routine in Chapter IV is applied to recover the integrality and nonpreemptiveness of the LP solutions.

First let decision variable y_{ijt} be defined as follows:

$$y_{ijt}: \begin{cases} 0 < y_{ijt} \le 1, \text{ if job } j \text{ is being processed on machine } i \text{ in the time interval } [t, t+1]. \\ 0, \text{ otherwise.} \end{cases}, t = 1,...,T.$$

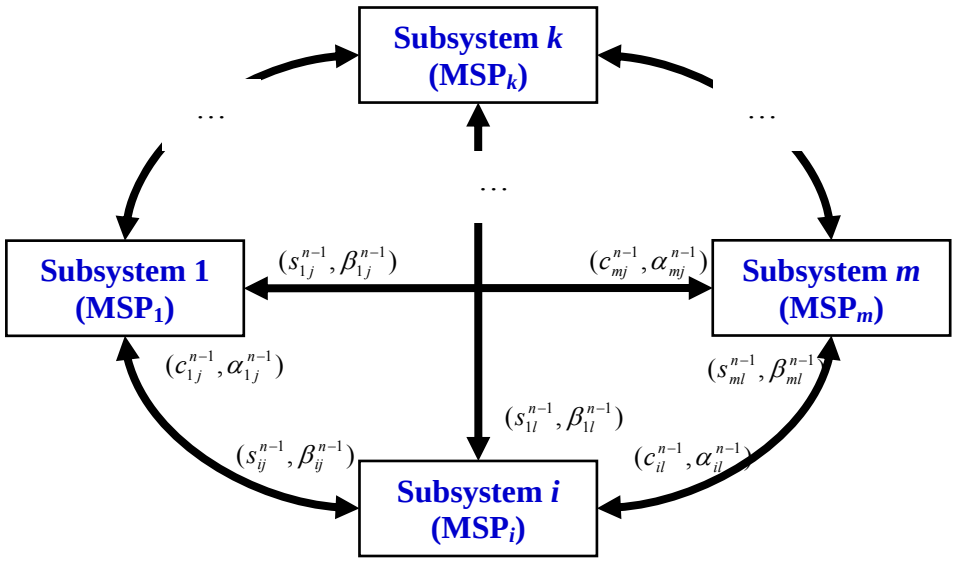

Figure 8 Information flows of DIPIS model for a distributed jobshop problem with *m* subsystems.

Additionally, the following notations are used for the mathematical formulation of the problem:

r_{ij}^n : ready time of job j on machine i at nth iteration. $r_{ij}^n = c_{lj}^{n-1}$ where $(\mathrm{Olj}, \mathrm{Oij}) \in \mathrm{A_j}$

d_{ij}^n : due date of job j on machine i at nth iteration. $d_{ij}^n = s_{lj}^{n-1}$ where $(\mathrm{Oij}, \mathrm{Okj}) \in \mathrm{A_j}$

L_{ij}^n : lateness of job j on machine i at nth iteration. $L_{ij}^n = c_{ij}^n - d_{ij}^n = \Theta_{ij}^n - \Phi_{ij}^n$

Φ_{ij}^n : earliness of job j on machine i at nth iteration. $\Phi_{ij}^n = \max(0, d_{ij}^n - c_{ij}^n)$

Θ_{ij}^n : tardiness of job j on machine i at nth iteration. $\Theta_{\mathrm{ij}}^{\mathrm{n}} = \max(0, c_{ij}^n - d_{ij}^n)$

Then, the machine problem of subsystem i (**MSP$_i$**) is proposed as follows:

$$\textbf{(MSP}_i\textbf{):}\quad \text{Min} \quad \sum_{j=1}^{N} w_j (c_{ij}^n - r_{ij}^n) + \sum_{j=1}^{N} \beta_{kj}^{n-1} \Theta_{ij}^n \tag{5.9}$$

$$\text{St.} \quad c_{ij}^n - \frac{1}{p_{ij}} \sum_{t=1}^{T} (t + \frac{1}{2}) y_{ijt} + \frac{p_{ij}}{2} = 0 \qquad j = 1, \ldots, N \tag{5.10}$$

$$c_{ij}^n \geq r_{ij}^n + p_{ij} \qquad \forall j \tag{5.11}$$

$$L_{ij}^{n} - c_{ij}^{n} = d_{ij}^{n} \qquad \forall j \qquad (5.12)$$

$$L_{ij}^{n} - \Theta_{ij}^{n} + \Phi_{ij}^{n} = 0 \qquad \forall j \qquad (5.13)$$

$$\sum_{t=1}^{T} y_{ijt}^{n} = p_{ij} \qquad \forall j \qquad (5.14)$$

$$\sum_{j=1}^{N} y_{ijt}^{n} \le 1 \qquad t = 1,...,T \qquad (5.15)$$

$$L_{ij}^{n} \text{ unrestricted} \qquad \forall j \qquad (5.16)$$

Note that, in the second term of the objective (5.9), β_{kj}^{n-1} is applied when subsystem *i* schedules job *j* on its machine later than the proposed due date of subsystem *k*, $d_{ij}^{n} = s_{kj}^{n-1}$, thus causing tardiness in the amount of $\Theta_{ij}^{n} = c_{ij}^{n} - s_{kj}^{n}$. Using the formula (5.10), the completion time of job *j* on the machine of subsystem *i* is calculated. If there is a nonpreemptive integral schedule of job *j*, (5.10) provides the exact completion time. If it is preemptive, (5.10) provides the average completion time of the schedule. Constraint (5.11) ensures that job *j* be scheduled after the previous operation of job *j* is finished processing. (5.12) and (5.13) are used to determine tardiness. Through constraint (5.14), job *j* is sure to be finished within the planning horizon. Constraint (5.15) implies that the maximum available machine capacity per time slot is one. (5.16) means that lateness can take a positive, zero, or negative value.

Suppose a central decision maker exists to apply traditional Lagrangian relaxation for solving the current problem. Then, by relaxing the precedence constraints, the problem can be decomposed into *m* machine subproblems. The objective of the LR subproblem would be as follows:

$$\text{Min} \quad \sum_{j=1}^{N} w_j (c_{ij} - r_{ij}) - \sum_{j=1}^{N} \theta_{kj} (s_{kj} - c_{ij}) \qquad (5.17)$$

s_{kj} is the starting time of the operation *k* of job *j* on machine *k* where (Oij, Okj) $\in A_j$. The second term is the function used to penalize the precedence violation if Oij is scheduled later than the proposed completion time of Okj. However, in a distributed

system environment there is no central entity that can update Lagrangian multiplier θ_{kj} using a global upperbound. Simply, subsystem *i* does not have full information about Okj and cannot control the schedule of Okj which belongs to subsystem *k*. DIPIS proposes a new term that can be represented by the variables of subsystem *i* using partial information received from subsystem k, as shown in (5.9). Detailed derivation of the penalty weights associated with Lagrangian multipliers will be explained in the following section.

After solving **MSP*i***, the feasibility restoration routine introduced in Chapter IV is applied to obtain a non- preemptive integral schedule from the LP solution.

3.2. *Derivation of the penalty weights*

The local objective of subsystem *i*, (5.9), can be restated using the information received from subsystem *k*, $(s_{kj}^{n-1}, \beta_{kj}^{n-1})$, as follows:

$$\sum_{j=1}^{N} w_j (c_{ij}^n - r_{ij}^n) + \sum_{j=1}^{N} \beta_{kj}^{n-1} \Theta_{ij}^n = \sum_{j=1}^{N} w_j (c_{ij}^n - r_{ij}^n) - \sum_{j=1}^{N} \beta_{kj}^{n-1} (s_{kj}^{n-1} - c_{ij}^n) \qquad (5.18)$$

The first term of the objective is a decomposed part of the global objective associated with only the operations of subsystem *i*. The second term is introduced to minimize the precedence violations between the operations that belong to different subsystems. So, β_{kj}^{n-1} should reflect the variation of the function $-\sum_{j=1}^{N} \theta_{kj} (s_{kj} - c_{ij})$ for the violations occurred when subsystem *i* schedules Oij tardier than the proposed starting time of Okj, i.e. $c_{ij}^n > s_{kj}^{n-1}$. Note that in a distributed system environment, β_{kj}^{n-1} can only be approximated using partial local information at $(n\text{-}1)^{th}$ iteration. Therefore, the tardiness weight β_{kj}^{n-1} is proposed as follows:

$$\beta_{kj}^{n-1} = \left. \frac{\partial \theta_{kj}^{n-1} (s_{kj} - c_{ij})}{\partial s_{kj}} \right|_{s_{kj} = s_{kj}^{n-1*}} = \theta_{kj}^{n-1} \qquad (5.19)$$

3.3. *Updating the Lagrangian multiplier for the subsystems*

Section 3.3 describes how to update Lagrangian multipliers using only the partial information received from other subsystems.

Let θ_{ij}^n be the Lagrangian multiplier for the precedence constraints between the pair of operations, (Olj, Oij), of job j at n^{th} iteration which is updated by subsystem i. Given $(s_{kj}^{n-1}, \beta_{kj}^{n-1})$, the subgradient optimization method is modified to calculate the Lagrangian multipliers using only partial information:

$$\theta_{ij}^n = \max(0, \theta_{ij}^{n-1} - \tau_i^n (s_{kj}^{n-1} - c_{ij}^n)) \tag{5.20}$$

$$\tau_i^n = \frac{\psi_i^n \left| Z_i^n - Z_{i,UB}^n \right|}{\sum_{j=1}^{N} \left(s_{kj}^{n-1} - c_{ij}^n \right)^2} \tag{5.21}$$

$$\begin{aligned} \psi_i^n &= \psi_i^{n-1} \times p, \quad \text{if } \left| Z_i^n - Z_{i,UB}^n \right| > \left| Z_i^{n-1} - Z_{i,UB}^{n-1} \right| \\ &= \psi_i^{n-1} \qquad \text{Otherwise} \end{aligned} \tag{5.22}$$

where Z_i^n is the local objective value of subsystem i problem at n^{th} iteration and $Z_{i,UB}^n$ is the local objective value of an upperbound solution of subsystem i problem obtained through the following routine.

Upperbound Calculation Routine.

Initialization. Identify the processing sequence of jobs from the schedule at n^{th} iteration.

Step 1. Inform completion times of the operations to the subsystems associated with the precedence constraints.

Step 2. Using the completion times received from other subsystems as new ready times of the associated operations, check if the current schedule has any ready time violations. If so, build a new schedule that observes the processing sequence identified at initialization. Otherwise, keep the current schedule.

Step 3. If the current schedule does not violate any precedence constraints, check if other subsystems also have schedules without ready time violations. If so, stop here and

calculate the local objective value with the current schedule. Otherwise, go to step 4.

Step 4. Check the cycle problem of the schedule. If the completion time of the last operation in the processing sequence becomes greater than the planning horizon, it implies that some operations of current processing sequence cause a cycle (or cycles). Then, stop here and use $Z_{i,UB}^{n-1}$ as $Z_{i,UB}^{n}$. Otherwise, go to step 1.

From the routine described above, we can get a globally feasible local schedule even though the objective value may not be good. Note that subsystems still exchange partial local information for this purpose.

3.4. *DIPIS algorithm for distributed jobshop scheduling problems*

In this section, DIPIS algorithm for distributed jobshop scheduling problems is summarized.

Initialization: Set the number of maximum interaction *N*.

Set, $\beta_{kj}^{0} = 0$, $\tau_i^0 = 0$, $\psi_i^0 = 2$, $\forall k, j$ where $k = 1,\ldots,m, j = 1,\ldots,N$

and p, $0 < p < 1$. Set $n = 1$.

Step 1: Sub-production system's problem. For $i = 1,\ldots,m$.

Step 1.1: Solve the problem **MSP*i*** and apply feasibility restoreation routine to find s_{ij}^{n} , c_{ij}^{n} .

Step 1.2: Calculate the step length ψ_i^n as shown in (5.22).

Step 1.3: Calculate the step size τ_i^n as shown in (5.21).

Step 1.4: Update the Lagrangian multipliers θ_{ij}^{n} as shown in (5.20) $\forall j$.

Step 1.5: Calculate penalty weights β_{ij}^{n}, $\forall j$ as shown in (5.19).

Step 1.6: Distribute the information vector $(s_{ij}^{n}, \beta_{ij}^{n})$ to subsystem *l*, where (Olj,Oij) $\in A_j$ and $(c_{ij}^{n}, 0)$ to subsystem *k*, (Oij, Okj) $\in A_j$. $\forall j$.

Step 2: If $n = N$ or $c_{ij}^{n} = c_{ij}^{n-1}$ for $\forall i, j$, stop. Otherwise, $n = n + 1$ and go to step 1.

The solution procedure starts with the initialization: setting all the penalty weights equal to zero and other parameters to starting values. At every iteration, each subsystem solves its local problem and updates the penalty weights, as shown in step1. For the penalty weight updates, the subsystems need a cooperative solution which will not violate the precedence constraints when applied to them. The cooperative solution of subsystem i at n^{th} iteration can be calculated using the upperbound calculation routine in section 3.3. The objective value of this solution is used as $Z^{n}_{i,UB}$ for updating step size and step length (step 1.2 and 1.3).

To calculate step size (step 1.3), precedence violations are quantified as the sum of difference between the completion time of the operations in the current subsystem and the starting time of their successors, i.e. $\sum_{j=1}^{N}\left(s_{kj}^{n-1} - c_{ij}^{n}\right)^2$ for (Oij, Okj) $\in A_j, \forall j$. For each of the precedence constraints, the associated Lagrangian multiplers are updated using the step size (step 1.4). The penalty weights are determined using the Lagrangian multipliers (step 1.5). Then, the information vectors in the prescribed format are sent to the coupled subsystems (step 1.6).

The iteration continues until the algorithm satisfies the predetermined stopping criteria (step 2). In this research, two stopping criteria are applied: the algorithm stops when (i) it achieves a converged solution, (ii) the number of the direct interactions reaches a predetermined value, i.e. $n + 1 = N$. The converged solution means the completion times of the operations in all jobs do not change iteration by the iteration. If a converged solution does not violate any of the precedence constraints, it is a compromised solution; meaning, it is a globally feasible solution even if it may not be globally optimal.

4. Experimental results

This section explains the experimental study used to investigate the performance of DIPIS for solving the distributed jobshop scheduling problems. The algorithm is

implemented using C programming language. CPLEX is used to solve the subsystem problems in LP formulation.

As performance measures to evaluate the quality of DIPIS solutions, Percent Deviation (PD) and Precedence Violation (PV) are used. PD is calculated to examine the closeness of the solution after global feasibility restoration to the optimal solution, or the best solution if an optimal solution is not available.

$$\mathrm{PD} = \frac{\left|Z - Z^*\right|}{Z^*} \times 100(\%) \tag{5.23}$$

Here Z and Z^* are the global objective value of the DIPIS solution and the optimal solution, respectively.

Subsystems solve machine problems with relaxed precedence constraints using starting times and completion times received from other subsystems. Therefore the degree of precedence violations in subsystems' solutions before global feasibility restoration is a good indicator of performance.

$$\mathrm{PV} = \frac{\sum_{\forall j} \max\left(0, \left(c_{ij} - s_{kj}\right)\right)}{T} \times 100(\%) \qquad \text{where (Oij, Okj)} \in \mathrm{A_j} \tag{5.24}$$

Finally, computation times of DIPIS are presented to check the efficiency of the algorithm.

4.1. *Experiments*

This study describes the experimental results of DIPIS for solving jobshop scheduling problems under partial information. DIPIS solutions are compared to the non-delay schedule generated by applying the WSPT rule given all the job processing times for job priorities. The jobshop problems are NP-hard, thus difficult to get the optimals. Alternatively, WSPT results are used for the comparison in this experimental study. WSPT rule is known to work well for a scheduling problem with the objective of total weighted completion times (Pinedo, 1995).

The jobshop problem of scheduling twenty jobs on five machines is tested for investigating the performance of the proposed algorithm. The problem data are generated as follows:

(i) Weights are randomly generated from uniform distribution U(1,20).

(ii) The processing times are randomly generated from U(1,100).

Two types of problems are examined. The problem types are classified by the characteristic of the operation sequences. Problem type 1 (P1) has a special routing structure: machines are divided into two groups. The machines in group one are designated for the first two operations of jobs and the ones in the other group for the later three operations. In problem type 2 (P2), the operation sequences of jobs is randomly generated, i.e. the operations of each order are randomly distributed to all machines. Ten instances are generated and tested for each type.

Table 7 shows the PD results of DIPIS algorithm. Due to the complexity of jobshop problems, optimal solutions of the test problems are not available. Instead, PD measures the closeness of the solution to the WSPT nondelay solution. The results show that DIPIS has the solutions with less than 6% of average deviations. In some problem instances, DIPIS generates better solutions than WSPT.

Two hypothesis tests are performed to identify if the variations in operation sequences affect the performance of the proposed algorithm. The first test hypotheses are as follows:

$$H_0\text{: } \mu_{PD,DIPIS,P1} = \mu_{PD,WSPT,P1} \text{ and } H_1\text{: } \mu_{PD,DIPIS,P1} > \mu_{PD,WSPT,P1}$$

Here $\mu_{PD,DIPIS,P1}$ and $\mu_{PD,WSPT,P1}$ denote the average PDs of DIPIS and WSPT when solving P1 instances, respectively. At significance level of 5 %, H_0 is not rejected with p-value equal to 0.2688. This means there is no difference in the average PDs between the two algorithms.

The next test hypotheses are as follows:

$$H_0\text{: } \mu_{PD,DIPIS,P2} = \mu_{PD,WSPT,P2} \text{ and } H_1\text{: } \mu_{PD,DIPIS,P2} > \mu_{PD,WSPT,P2}$$

Here $\mu_{PD,DIPIS,P2}$ and $\mu_{PD,WSPT,P2}$ denote the average PDs of DIPIS and WSPT when solving P2 instances, respectively. At significance level of 5 %, H_0 is rejected with p-value equal to 0.0389. This tells us that, for the random routing problems, WSPT works better than DIPIS.

Table 7 also shows the PV results which represent the degree of precedence violations. For both problem types, PV is less than 2% which is considered small.

Table 8 reports the computational times of DIPIS for solving the jobshop scheduling problems. A statistical test is performed to check if the variations in the operation sequence affect the computational times of DIPIS. The hypotheses are as follows:

$$H_0: \ \mu_{Comp.Time,P1} = \mu_{Comp.Time,P2} \text{ and } H_1: \ \mu_{Comp.Time,P1} > \mu_{Comp.Time,P2}$$

At significance level of 5 %, H_0 is not rejected. The result means that the difference between the average computational times for two problem types is not statistically significant (p-value = 0.7693). Therefore, the computational efficiency of the proposed algorithm is not affected by the difference in the operation sequence types.

5. Conclusions

In this chapter, DIPIS is applied to jobshop scheduling problems in a distributed system consisting of multiple subsystems. Each subsystem is responsible for the scheduling of jobs on its own machine. The local processing information of each subsystem is considered private and is not exposed to other subsystems. Using direct interaction among subsystems, partial information is exchanged for the optimization of the global objective (represented by the sum of the subsystems' local objectives). The experiment results show that DIPIS generates competitive solutions within less than 6% worse than WSPT results. Considering that the proposed algorithm solves the problems under conditions of partial information sharing and no central coordination, the performance of DIPIS is encouraging.

Table 7 PD and PV results of DIPIS algorithm for Jobshop problems

PROBLEM TYPE	DIPIS algorithm					
	PD (%)			PV (%)		
	Min	Avg	Max	Min	Avg	Max
P1	-4.11	2.13	11.63	0	0.76	7.58
P2	-4.72	5.37	22.02	0	1.57	9.90

Table 8 Computational results of the DIPIS algorithm

PROBLEM TYPE	DIPIS algorithm		
	COMPUTATIONAL TIMES (SEC.)		
	MIN.	AVG.	MAX.
P1	802.33	1031.74	1753.28
P2	860.64	1063.66	1333.70

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH

1. Conclusions

The purpose of this dissertation is to develop a general solution methodology for solving optimization problems in a distributed system environment where decision making authorities and system information are distributed among multiple subsystems. No central decision making activity is involved in the optimization procedure. Instead, the subsystems cooperate to achieve the system goal as well as the local subsystem goals. The local information of the subsystems is not fully exposed to one another but *partial information sharing* occurs among subsystems for system optimization.

Direct Interaction for Partial Information Sharing (DIPIS) is proposed to model the aforementioned distributed system and solve the associated optimization problems. Direct interactions are established among subsystems for partial information sharing associated with coupling constraints in order to find a compromised system solution. The proposed solution method is based on Lagrangian Relaxation method modified to work without global system control or complete system information. Chapter II describes the mathematical model of DIPIS, interaction mechanism for partial information sharing, and the information format. Detailed solution procedures are explained with the associated formulas.

In Chapter III, distributed capacity allocation problems are explored as an application of DIPIS. The purpose of this study is to investigate the performance of the proposed method for solving problems in LP model. The convex combination rule proposed by Choi and Kim (1999) is used to resolve the issue of the solution oscillation in LP model. The experimental results show that DIPIS outperforms CICA and LR in the distributed capacity allocation problems with two subsystems and one facility. Even though more extensive experiments involving an increased number of subsystems is necessary, the

results are encouraging; DIPIS performs competitively under partial information without central or third party coordination for achieving the global goal.

Single-machine distributed scheduling problems are presented in Chapter IV as the second application. Two types of mathematical formulations are introduced: IP and LP. The IP model, as previously pointed out in CICA model, reveals a limitation in solving the problems of large size; i.e. as the number of jobs and the processing times increase, the computational times increase exponentially. The main contribution of this chapter is the development of the heuristic based on LP model. The results show that the performance of DIPIS with LP heuristic is promising. Computational times are considerably smaller compared to the centralized algorithm while maintaining the solution quality within on average 6 % from the optimal.

The final application of the research is the distributed jobshop scheduling problem. Whereas the proposed solution model appears to be similar to the traditional machine decomposition method, the local job processing information on each machine in DIPIS model is independently managed without being fully exposed to any other entity. For a globally feasible schedule, partial solutions and the associated weights are exchanged in predetermined format. In the experiments, two types of problems classified by the characteristic of the operation sequences are tested and compared with a centralized WSPT non-delay schedule. The results indicate that DIPIS performs well by generating the solutions within 6 % of average deviation from WSPT values.

2. Future research

This section presents other issues in the current research and the direction of future research.

In this research, the subsystems participating in the global system optimization directly interact for partial information sharing by the predetermined time schedule, which is called *synchronous interaction*. However, this type of interaction may not fit all the real-world cases. As the number of subsystems increases, the degree of the distributedness of the global system information and decision making authorities

increases as well; then, it would become more difficult to synchronize all the interactions among the participating subsystems because each subsystem manages different local tasks and schedules according to the local system condition. Enforcing synchronous interactions under this situation requires more restricted interaction rules, which may deteriorate the performance of subsystems and, in the worst case, the global system goal. Alternatively, a more flexible interaction method where any subsystem can initiate the interaction process in its preference would be more advantageous. This type of interaction is called *asynchronous interaction*. Therefore, asynchronous interaction is an interesting topic for future research.

In scheduling applications, we proposed LP heuristic to improve the computational efficiency of the solution method as an alternative of IP model. As expected, the heuristic was effective in reducing the computational times, especially for solving the larger problems. However, from the aspect of the solution quality, LP heuristic may not be considered superior to IP model. For example, if using IP, we can always find the local optimum, while LP heuristic provides an approximate solution. Even though the local optimum is not always favorable from the global viewpoint, it is still a good candidate of the global optimum. Another reason is that the heuristic may lose some of the system information due to the approximation steps. Therefore, the development of a model that can fully use the system information (as much as received) and then provide the best intermediate solution will be a desirable direction for future research.

The global feasibility of the distributed solution, as mentioned in Jeong (2001), is also an important issue for applying the proposed method to real-world problems. Global feasibility is difficult to achieve without the global view of the system or the centralized control of the system information.

REFERENCES

Benders, J. R. (1962) Partitioning procedure for solving mixed variables programming problems. *Numerische Mathematik*, 4, 238-252.

Bertsekas, D. P. (1988) The auction algorithm : a distributed relaxation method for the assignment problems. *Annals of Operations Research*, 14, 105-123.

Bertsekas, D. P. (1990) The auction algorithm for assignment and other network flow problems: a tutorial. *Interface*, 20(4), 133-149.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989) *Parallel And Distributed Computation: Numerical Methods*, Prentice-Hall., Englewood Cliffs, NJ.

Choi G. and Kim C. (1999) Primal recovery strategy for Lagrangian dual subgradient-based methods, *Proceedings of the 99' Spring Conference of The Korean Institute of Industrial Engineers /Korean Operation Research and Management Science Society*, Korea, 313-314.

Dantzig G. B. and Wolfe P. (1961) The decomposition algorithm for linear programs. *Econometrica*, 29, 767-778.

Davis, R. and Smith, R. G. (1983) Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20, 63-109.

Duffie, N. A. and Prabhu, V. V. (1994) Real-time distributed scheduling of heterarchical manufacturing systems. *Journal of Manufacturing Systems*, 13(2), 94-107.

Ertogral, K. and Wu S. D. (2000) Auction-theoretic coordination of production planning in the supply chain. *IIE Transactions*, 32, 931-940.

Fisher, M. (1981) The Lagrangian Relaxation method for solving integer programming problems. *Management Science*, 27, 1-18.

Fox, M. S. and Smith, S. F. (1984) ISIS-a knowledge-based system for factory scheduling. *Expert System*, 1(1) 25-48.

Gou, L., Hasegawa, T. and Luh, P. B. (1994) Holonic planning and scheduling for a robotic assembly Testbed, *Proceedings of the 4th Rensselaer International Conference on Computer Integrated Manufacturing and Automation Technology*, Troy, NY, 142-149.

Guo, Z., Koehler G. J., and Whinston A. B. (2002) Market-based optimization algorithms for distributed systems. Working paper, Department of Management Science and Information Systems, The University of Texas at Austin.

Jeong, I-J (2001) Cooperative interaction of distributed organizations via coupling agents: a methodology and applications. *Ph.D. Dissertation*. Industrial Engineering, Texas A&M University, College Station.

Jeong, I-J and Leon, V. J. (2002) Decision making and cooperative interaction via coupling agents in organizationally distributed systems. *IIE Transactions - Special Issue on Large Scale Optimization*, 34, 789-802.

Jeong, I-J and Leon, V. J. (2003) Distributed allocation of capacity of a single-facility using cooperative interaction via coupling agents. *International Journal of Production Research*, 1(1), 15-30.

Jeong, I-J and Leon, V. J. (2005) A single-machine distributed scheduling methodology using cooperative interaction via coupling agents. *IIE Transactions*, 37(2), 37-152.

Kate, A. T. (1972) Decomposition of linear program by direct distribution. *Econometrica*, 40(5), 883-898.

Kutanoglu, E. and Wu S. D. (1999) On combinatorial auction and Lagrangian Relaxation for distributed resource scheduling. *IIE Transactions*, 31(9), 813-826.

Lin, G. Y. and Solberg, J. J. (1992) Integrated shop floor control using autonomous agents. *IIE Transactions*, 24(3), 57-71.

Murphy L., Contreras J., and Wu F. (1995) A decomposition-coordination approach for large-scale optimization, *Proceedings SIAM Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, 78-83.

Pinedo, M. (1995) *Scheduling: Theory, Algorithms and Systems*, Prentice-Hall., Englewood Cliffs, NJ.

Roundy, R. D., Maxwell, W. L., Herer, Y. T., Tayur, S. R. and Getzler, A. W. (1991) A price-directed approach to real-time scheduling of manufacturing operations. *IIE Transactions*, 23, 149-160.

Sherali, H. and Choi, G. (1996) Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. *Operations Research Letters*, 19, 105-113.

Smith, S. F., Ow, P. S., Potvin, J. Y., Muscettola, N., and Matthys, D. C. (1990) An integrated framework for generating and revising factory schedule. *Journal of the Operations Research Society*, 41, 539-552.

Sycara, K., Roth, S., Sadeh N., and Fox M. S.(1991) Distributed constrained heuristic search. *IEEE Transactions On Systems Man and Cybernetics*, 21(6), 1446-1461.

Talukdar, S., Baerentzen L., Gove A. and de Souza, P. (1998) Asynchronous teams: cooperation schemes for autonomous agents. *Journal of Heuristics*, 4(4), 295-321.

Ure, J. and Jaegersberg, G. (2005) *Invisible Architecture -The Benefits Of Aligning People, Processes And Technology: Case Studies For System Designers And Managers*, British Computer Society, UK.

Wellman, M. P., Walsh, W. E., Wurman, P. R. and MacKie-Mason, J. K. (2001) Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35, 271-303.

APPENDIX

LP-CICA – HEURISTIC TO IMPROVE COMPUTATIONAL TIMES OF SINGLE-MACHINE DISTRIBUTED SCHEDULING PROBLEMS

The appexdix introduces LP-CICA, LP heuristic developed for CICA method to solve a single-machine distributed scheduling problem of large size in practical times.

A.1. *Introduction*

We consider a single machine scheduling problem where autonomous decision makers exist in each of multiple sub-production systems. Sub-production systems independently maintain their own jobs to schedule while communicating with each other to pursue a global goal of minimizing total weighted completion times of the jobs. The primary assumption is that none of the sub-production system and shared machine has complete knowledge about the entire system. Cooperative Interaction via Coupling Agents (CICA) methodology was proposed to apply to the described scheduling problems and tested with small size of jobs. Although CICA algorithm generated promising results, it also showed the limitation to deal with the problems with large size of jobs. Thus, we develop a heuristic to efficiently solve the established single machine scheduling problems, i.e. reduce the computational time in a practical range even for the problems of large job sets. The heuristic is developed based on LP relaxation formulation of the associated single machine scheduling problems, modifying to specifically address the current problem. The experimental results show significant improvement in computational time with still maintaining the solution qualities in reasonable levels.

A.2. *Motivation*

A distributed scheduling problem is a problem where the decision authorities and information are distributed in multiple sub-production systems and no complete

information sharing is achievable among the sub-production systems. Thus, centralized solution methodologies to optimize the problem with one complicated model are hardly applicable or not recommended due to the implementation issues such as unrealistic computational times to solve the problem. In lieu of this, alternative distributed solution methodologies have been actively studied among OR, Computer Science and Artificial Intelligence researchers [4, 10].

Cooperative Interaction via Coupling Agents (CICA) has been developed by Jeong and Leon [4] to deal with the distributed problems in an appropriate manner. CICA is based on Lagrangian Relaxation (LR) with modifications to reduce the amount of global information required for its application. CICA and LR differ in the way coupling constraints can be relaxed, and the information sharing associated with coupling constraints. Specifically, in the CICA model, decision agents termed Coupling Agents (CA) manage coupling constraints to reach compromise among the solutions of separable sub-problems which are associated with quasi-autonomous agents termed Coupled Autonomous Organizations (CAO). This is a major difference of CICA from LR in which all coupling constraints are dealt by a single decision entity [4].

In this paper, a heuristic, termed LP Relaxed CICA (LP-CICA), is proposed to solve problems of realistic size in practical times. A significant limitation of the CICA formulation proposed in Jeong and Leon [5] is that the computational times required to solve large problems prohibitively increase as the number of jobs increases. LP-CICA utilizes LP relaxation formulation proposed by Dyer and Wolsey [2] to apply CICA methodology to the current distributed scheduling problems. However, by applying CICA methodology to the scheduling problems with a linear function of completion times, the resultant problems turn out to be earliness/tardiness (E/T) scheduling problems. Consequently, in LP-CICA formulations, new constraints are inevitably introduced into the formulations proposed by Dyer and Wolsey [2] so that LP-CICA manages corresponding E/T scheduling problems in an appropriate manner.

The primary differences between CICA formulation and LP-CICA formulation are the integrality relaxation of the 0/1 decision variables and the relaxation of the non-

preemptive constraints. Thus, the performance of LP-CICA is expected to be better than CICA from the standpoint of the computational times.

A.3. *Single Machine Distributed Scheduling*

We consider a manufacturing facility that has *m* sub-production systems and a shared machine. Let *U* be the set of all jobs in the facility and U_i , a subset of *U*, that comprises jobs in sub-production system *i, where* $|U_i| = n_i$, $i = 1,...,m$. It is assumed that the job subsets are disjoint so that the shared machine processes total of $N = |U| = \sum_{i=1}^{m} n_i$ jobs. The goal in this study is to schedule *N* jobs on the shared machine to minimize total weighted completion times in a distributed manner. What meant by 'distributed' is that (1) there exists a decision maker in the sub-production system *i* who can authorize only the schedule of its n_i jobs and the relevant information, (2) each sub-production system and the shared machine can exchange only partial information (for a more complete description of this, the reader is referred to [4, 5, 10]).

Using notation similar to the one in Dyer and Wolsey [2] and Uma and Wein [14], let T denote the planning horizon and the decision variable x_{jt} be defined as,

$$x_{jt} = \begin{cases} 1, & \text{if job } j \text{ is being processed in the time } \textit{period } [t, t+1]. \\ 0, & \text{otherwise.} \end{cases}, t = 1,...,T.$$

Associated with job *j* are the processing time p_j, start time $s_j = \frac{1}{p_j}\sum_{t=1}^{T}(t+\frac{1}{2})x_{jt} - \frac{p_j}{2}$, completion time $C_j = s_j + p_j = \frac{1}{p_j}\sum_{t=1}^{T}(t+\frac{1}{2})x_{jt} + \frac{p_j}{2}$ and the weight per unit of completion time w_j.

The machine capacity constraints are introduced as $\sum_{j \in U} x_{jt} \le 1, \ t = 1,...,T$ so that the maximum number of jobs that can be processed in any time slot must be less than or equal to one. And the precedence constraints between jobs in the sub-production system are also considered in this problem. If job *q* precedes job *r* where $q \in U_i$ and $r \in U_i$, then the

precedence relation is shown as (*q, r*). Now the precedence constraints are represented as $C_r - C_q \geq p_r, \forall(q,r) : q,r \in U_i, \quad i = 1,...,m$.

As mentioned earlier, the problem of concern here is considered as distributed scheduling instance. Sub-production system *i* can recognize the partial objective function terms $\sum_{j \in U_i} w_j C_j$, and capacity constraints, precedence constraints associated with jobs $j \in U_i$. Naturally the schedule claimed by the sub-production system *i* may have machine conflict with schedules claimed by other sub-production systems $k \neq i$. Therefore, the main question is how to generate the schedule of jobs $j \in U$ on the shared machine without capacity violation under partial information sharing.

A.4. *LP relaxed CICA*

The problem is formulated by CICA model proposed by Jeong and Leon [4], as shown in Fig. A.1. Associated with the shared machine there is a problem **(MP)**, and associated with each sub-production system there is a problem **(SP*i*)**. Sub-production systems communicate with the shared machine by exchanging information triplets during an iterative process. At iteration *n*, the information from **(MP)** to **(SP*i*)** consists of a triplet formed by a solution vector of a scheduling problem s_{yj}^{n-1} , and weight vector μ_j^{n-1} , π_j^{n-1} that represent the starting time of job *j* determined by the shared machine, and the costs of shifting job *j* one unit left/right from its position in s_{yj}^{n-1} , respectively. Similarly, the information from **(SP*i*)** to **(MP)** consists of a solution vector s_{xj}^{n} , weight vector α_j^n , β_j^n , determined by the sub-production system. For further information on CICA solution methodology, the reader is referred to [4, 5].

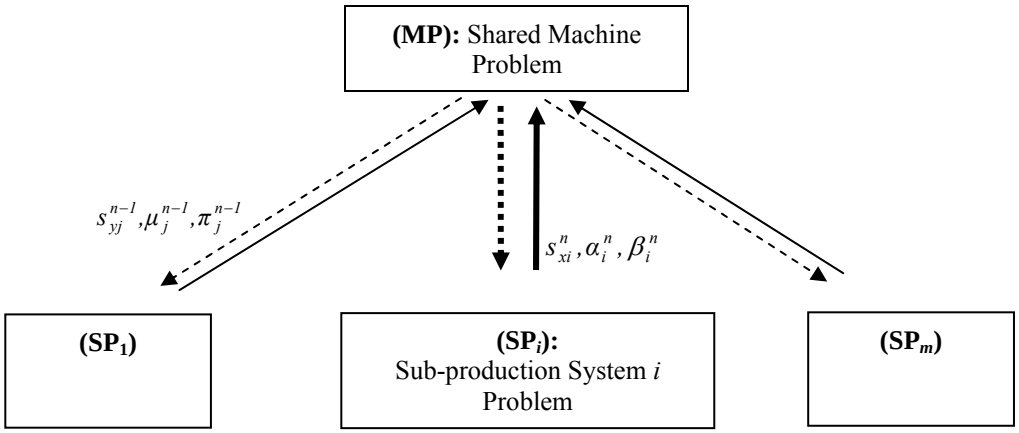

Fig. A 1 CICA model for a single machine problem [5]

The following two sections describe the problem formulations for the shared machine and the sub-production systems. Section A.4.1 describes LP relaxed CICA and its feasibility restoration procedure to recover the integrality and non-preemptiveness of the solution.

A.4.1. *CICA formulation for sub-production system*

Let s_{yj}^{n-1} and C_{yj}^{n-1} be the starting time and the completion time of job *j* determined by the shared machine after $(n\text{-}1)^{th}$ iteration, respectively. Note the subscript *y* indicates that the information is specified by the shared machine. μ_j^{n-1} and π_j^{n-1} are the earliness/tardiness (E/T) weights by starting job j one unit early/late from s_{yj}^{n-1}. Then the problem of sub-production system *i* (**SP*i***) at n^{th} iteration can be formulated as follows:

$$\textbf{SP}_i\text{: Min} \quad \sum_{j\in U_i} w_j C_j + \sum_{j\in U_i} \mu_j^{n-1} E_j^n + \sum_{j\in U_i} \pi_j^{n-1} \Gamma_j^n \tag{A.1}$$

$$\text{St.} \quad \sum_{t=1}^{T} x_{jt} = p_j, \qquad j \in U_i \tag{A.2}$$

$$\sum_{j \in U_i} x_{jt} \le 1, \qquad t = 1,...,T \qquad \text{(A.3)}$$

$$C_r - C_q \ge p_r, \qquad \forall (q,r) : q,r \in U_i \qquad \text{(A.4)}$$

$$C_j = \frac{p_j}{2} + \frac{1}{p_j} \sum_{t=1}^{T} (t + \frac{1}{2}) x_{jt}, \qquad \forall j \in U_i \qquad \text{(A.5)}$$

$$C_j - C_{yj}^{n-1} = \Gamma_j^n - E_j^n, \qquad \forall j \in U_i \qquad \text{(A.6)}$$

$$E_j^n - T \cdot \Theta_j \le 0, \qquad \forall j \in U_i \qquad \text{(A.7)}$$

$$\Gamma_j^n + T \cdot \Theta_j \le T, \qquad \forall j \in U_i \qquad \text{(A.8)}$$

$$C_j \ge p_j, \qquad \forall j \in U_i \qquad \text{(A.9)}$$

$$C_j \le T, \qquad \forall j \in U_i \qquad \text{(A.10)}$$

$$\Theta_j \in \{0,1\}, \qquad \forall j \in U_i \qquad \text{(A.11)}$$

$$x_{jt} \in \{0,1\}, \qquad \forall j \in U_i, \forall t \qquad \text{(A.12)}$$

Note that the objective function of the sub-production system *i* (A.1) is to find the best local solution along with minimizing deviation from the shared machine solutions. Constraint (A.2) is a typical scheduling constraint which implies all jobs must be finished within the planning horizon. Constraint (A.3) and (A.4) are the local capacity and precedence constraint, respectively. Of particular concern is constraint (A.5) which is to calculate the completion time of each job using decision variables [2]. This constraint is essential in LP-CICA because it can also provide an approximation of completion times even with a preemptive non-integer schedule. Constraints (A.6), (A.7), (A.8), (A.9), (A.10) and (A.11) are introduced to customize the formulation to the imposed E/T scheduling problems. Constraint (A.6) is to calculate E/T amount in case that the solution of the sub-production systems cannot match the shared machine one. Adding constraint (A.7) and (A.8) to the formulation ensures that earliness and tardiness cannot be positive at the same time for any job ($E_j^n \cdot \Gamma_j^n = 0, j \in U_i$). Constraint (A.9) and (A.10) are the feasibility constraints that enforce all jobs to be scheduled within the

planning horizon. Constraint (A.11) is 0/1 integer variables that are auxiliary to the constraints (A.7) and (A.8). Constraint (A.12) is 0/1 decision variable constraints which will be relaxed in LP-CICA.

A.4.2. *CICA formulation for shared machine*

The shared machine collects information from the sub-production systems. Let s_{xj}^n and C_{xj}^n be the starting time and the completion time of job *j* determined by the sub-production systems after n^{th} iteration, respectively. Note the subscript *x* indicates that the information is specified by the sub-production systems. α_j^n and β_j^n are the earliness/tardiness (E/T) weights by starting job j one unit early/late from s_{xj}^n. For the clarity of the formulation, y_{jt} is introduced as the decision variables of the shared machine problem. Then the shared machine problem (**MP*j***) can be formulated as follows:

$$\mathbf{MP_j}: \text{ Min } \quad \sum_{j\in U}(\alpha_j^n E_j^n + \beta_j^n \Gamma_j^n) \tag{A.13}$$

$$\text{St. } \quad \sum_{t}^{T} y_{jt}^n = p_j, \qquad \forall j \in U \tag{A.14}$$

$$\sum_{j\in U} y_{jt}^n \le 1, \qquad t = 1,...,T \tag{A.15}$$

$$C_j = \frac{p_j}{2} + \frac{1}{p_j}\sum_{t=1}^{T}(t+\frac{1}{2})y_{jt}, \qquad \forall j \in U \tag{A.16}$$

$$C_j - C_{xj}^n = \Gamma_j^n - E_j^n, \qquad \forall j \in U \tag{A.17}$$

$$E_j^n - T\cdot\Theta_j \le 0, \qquad \forall j \in U \tag{A.18}$$

$$\Gamma_j^n + T\cdot\Theta_j \le T, \qquad \forall j \in U \tag{A.19}$$

$$C_j \ge p_j, \qquad \forall j \in U \tag{A.20}$$

$$C_j \le T, \qquad \forall j \in U \tag{A.21}$$

$$\Theta_j \in \{0,1\}, \qquad \forall j \in U \qquad \text{(A.22)}$$

$$y_{jt} \in \{0,1\}, \qquad \forall j \in U, \forall t \qquad \text{(A.23)}$$

The formulation for shared machine problems are the same as one for sub-production systems as shown in the previous section except that (i) in the objective functions of the shared machine problems, total weighted completion times are not included, and (ii) the precedence constraints are excluded. The major concern of the shared machine problem is to make compromise among sub-production systems' schedules, not to minimize the completion times of jobs. And the precedence relationship between jobs is considered to be private information of the sub-production systems. Therefore, it is not known to the shared machine in the distributed system environment under this study.

A.4.3. *LP relaxed CICA*

The integrality of binary decision variables are relaxed in the formulations for both sub-production systems and shared machine. That is, decision variables, $x_{jt,}$ have fractional values between zero and one. Undoubtedly, the resulted solutions are not guaranteed to be integral. Thus, an additional procedure to recover the feasibility of the solution is required at each iteration.

In this paper, a simple post-processing routine for the feasibility restoration is applied as follows:

Feasibility Restoration Procedure for LP-CICA

Step 1. Sort jobs in the ascending order of the completion time.

Step 2. If two or more jobs have the same completion time, order them by WSPT rule. If jobs are still tied, order them arbitrarily.

Step 3. Check the schedule overlapping between adjacent jobs. If overlapping occurs, reset the starting time of the latter job by the completion time of the preceded job.

Step 4. Check if the last job finishes processing within the planning horizon. If not, readjust the completion time of the last job so that the last job finishes by the planning horizon. If overlapping occurs due to this adjustment, reset the completion time of the preceded job by the starting time of the latter job.

A.5. *Experimental study*

In this section, the performance of LP-CICA is experimentally compared with CICA and Lagrangian Relaxation using randomly generated problems. In Lagrangian Relaxation, the machine conflict constraints are relaxed and the centralized problem is decomposed into *m* independent sub-problems. The experiments are performed for two distributed environment settings, two sub-production systems with one shared machine and four sub-production systems with one shared machine. The algorithm is coded by C language and CPLEX version 8.1 is used as an optimization tool to solve the sub-production system problems and the shared machine problem.

For the performance measures of the algorithms, 'PD' reflecting the quality of the solution and the computational times are used. Although main interest in this paper is the improvement of computational times of the LP-CICA compared to CICA, the solution quality of LP-CICA algorithm is undoubtedly important for the algorithm evaluation. The closeness of the algorithm solution to the centralized solution is evaluated as follows:

$$\text{PD} = \frac{\left|Z\text{-}Z^*\right|}{Z^*}$$

where Z is the algorithm solution and Z* is the centralized solution [4,5].

The first experiment considers two sub-production systems with one shared machine. The weight for the completion time is generated from the discrete uniform distribution U(1,10). Two factors are considered in this experiment, the variance of the processing times and the number of jobs that have precedence constraints. Processing times are generated from either U(1,5) or U(1,10). And three levels of the precedence constraints are tested. As an illustrated example, consider the problem instance that each sub-

production system has 18 jobs to schedule. In level 1, four jobs have a precedence relationship (e.g., $(q,r) \in \{(1,2),(2,3),(3,4)\}$). In levels 2 and 3, nine and thirteen jobs are interrelated by precedence constraints respectively. Therefore, the total number of problem types is 6 and 10 problem replicates are generated for each type. The step parameter for CICA is set to 0.75 and the one for Lagrangian Relaxation is set to 0.5 because those parameter values are known to perform well empirically for each algorithm [3, 5]. The various problem sizes are tested so that the behavior of the algorithms under practical job settings can be observed.

Table A.1 and A.2 show the percent deviations of the solutions after global feasibility restoration and CPU times of Lagrangian relaxation, CICA and LP-CICA for various sizes of jobs. LP-CICA as well as Lagrangian relaxation and CICA yields solutions within 5 % deviation from optimal solutions, yet requiring less than half the CPU times of CICA algorithm for large jobs.

The second experiment considers four sub-production systems with one shared machine to investigate the behavior of the algorithm in multiple sub-production systems environments. The test problems are generated in the same method as the first experiment. The step parameter settings are also the same as the first experiment. To observe the performance of LP-CICA in moderately large problem size, problems of scheduling 36 jobs are tested.

As seen in Table A.3, the solution qualities of both CICA and LP-CICA somewhat deteriorate compared to two sub-production system environment setting while Lagrangian relaxation shows no significant change. Nevertheless, both algorithms still maintain solutions of less than 5 % percent deviations. Considering no complete information sharing among sub-production systems, performances of both algorithms in this setting are considered to be in tolerable level.

Table A.1 Percent Deviation comparison for large job sets

No. of Jobs	Percent Deviation								
	Lagrangian Relaxation			CICA algorithm			LP-CICA algorithm		
	min	avg	max	min	avg	max	min	avg	max
12	0.000	0.001	0.046	0.000	0.015	0.229	0.000	0.029	0.216
24	0.000	0.004	0.059	0.000	0.016	0.163	0.000	0.016	0.131
36	0.000*	0.000*	0.003*	0.000	0.012	0.123	0.000	0.015	0.112
48	-	-	-	-	-	-	0.000	0.014	0.055

*Since Lagrangian relaxation method requires significantly large computational times to solve the problems for 36 jobs, the LR results for 36 jobs are obtained from 12 test problems randomly generated from processing times of U(1,5) and precedence constraints of level 1.

Table A.2 Computational time comparison for large job sets

No. of Jobs	Computational Time (sec.)								
	Lagrangian Relaxation			CICA algorithm			LP-CICA algorithm		
	min	avg	max	min	avg	max	Min	avg	max
12	12.69	63.85	204.03	23.92	38.49	155.05	27.77	31.41	36.88
24	114.78	1541.40	9129.61	34.89	141.98	1280.61	40.83	61.15	99.30
36	1813.11*	7054.66*	13230.67*	69.67	1079.04	5446.33	75.25	132.52	237.56
48	-	-	-	-	-	-	123.09	301.37	690.69

*Since Lagrangian relaxation method requires significantly large computational times to solve the problems for 36 jobs, the LR results for 36 jobs are obtained from 12 test problems randomly generated from processing times of U(1,5) and precedence constraints of level 1.

Table A.3 Results comparison for 4 sub-production system problems

	Lagrangian Relaxation			CICA algorithm			LP-CICA algorithm		
No. of Jobs	Percent Deviation								
	min	avg	max	min	avg	max	Min	avg	max
36	0.000	0.001	0.004	0.000	0.035	0.245	0.000	0.037	0.281
No. of Jobs	Computational Time (sec.)								
	min	avg	max	min	avg	max	Min	avg	max
36	277.64	1092.79	3695.39	76.08	332.14	1507.78	81.20	133.12	209.92

A.6. *Conclusion*

In this study, in order to efficiently solve the established single machine scheduling problems, i.e. reduce the computational time in a practical range even for the problems of large job sets, we developed LP-CICA model in which the LP relaxation is applied on 0/1 integer decision variables.

The experimental results show that LP-CICA can reduce the computational times up to 5% of Lagrangian relaxation and 13 % of CICA while it obtain reasonable solution quality as explained in the previous section. The environment with multiple sub-production systems is also examined. LP-CICA can reduce the computational times up to 13% of Lagrangian relaxation and 40 % of CICA.

A.7. *References*

[1] Bazaraa, M. S., Sherali, H. D. and Shetty, C. M. (1993) *Nonlinear programming: Theory and algorithms*. 2nd Edn., John Wiley & Sons, New York

[2] Dyer, M. E. and Wolsey, L. A. (1990) Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, **26**, 255 - 270

[3] Fisher, M. (1981) The Lagrangian relaxation method for solving integer programming problems. *Management Science*, **27**, 1-18

[4] Jeong I. J. and Leon V. J. (2002) Decision making and cooperative interaction via coupling agents in organizationally distributed system. *IIE Transactions – Special Issue in Large Scale Optimization*, 34,789-802

[5] Jeong I. J. and Leon V. J. (2005) A single-machine distributed-scheduling methodology using cooperative-interaction via coupling-agents, *IIE Transactions*, Vol. 37, No.2, 137-152

[6] Nemhauser, G. L. and Wolsey, L. A. (1989) *Integer and combinatorial optimization*, John Wiley & Sons, New York

[7] Pinedo, M. (1995) *Scheduling: Theory, algorithms and systems*. Prentice Hall, Englewood Cliffs, NJ

[8] Pritsker A., Watters, L. and Wolfe P. (1969) Multiproject scheduling with limited resources: a zero-one programming approach. *Management Science: Theory*, **16**(1), 93-108

[9] Sherali, H. and Choi, G. (1996) Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. *Operations Research Letters*, **19**,105-113

[10] Sycara, K., Roth, S., Sadeh, N. and Fox, M. (1991) Distributed constrained heuristic search. *IEEE Transaction on Systems Man and Cybernetics*, **21**(6), 1446 – 1461

[11] Uma, R. N. and Wein, J. (1998) On the relationship between combinatorial and LP-based approaches to NP-hard scheduling. *Lecture Notes in Computer Science,* **1412**, 394-408

VITA

Sun Woo Kim received his B.S. degree in Mechanical Engineering from Yonsei University, Seoul, Korea, in 1991. After military service, for an M.S. degree, he entered the same department in 1994. While in the M.S. program, his research was focused on concurrent engineering and computer-aided process planning in CAD/CAM lab under the guidance of Prof. Soo-Hong Lee. He received his M.S. degree in 1996. The research fields of his interest include distributed optimization, production planning and scheduling, and manufacturing system analysis and control. His address in Korea is 269-12, Sadang 4 dong, Dongjak-gu, Seoul 156-094, Korea.