Dynamic Multi-Linked Negotiations in Multi-Echelon Production Scheduling Networks

Hoong Chuin LAU
School of Information Systems
Singapore Management University
hclau@smu.edu.sg

Guan Li SOH, Wee Chong WAN
The Logistics Institute – Asia Pacific
National University of Singapore
<tlisgl, tliwwc>@nus.edu.sg

Abstract

In this paper, we are concerned with scheduling resources in a multi-tier production/logistics system for multi-indenture goods. Unlike classical production scheduling problems, the problem we study is concerned with local utilities which are private. We present an agent model and investigate an efficient scheme for handling multi-linked agent negotiations. With this scheme we attempt to overcome the drawbacks of sequential negotiations and negotiation parameter settings. Our approach is based on embedding a credit-based negotiation protocol within a local search scheduling algorithm. We demonstrate the computational efficiency and effectiveness of the approach in solving a real-life dynamic production scheduling problem which balances between global production cost and local utilities within the facilities.

1. Introduction

In a multi-agent system, agents negotiate to find a mutually acceptable solution to a problem. Most often than not, agent negotiations are performed independently of one another. In this paper, we apply the concept of *multi-linked negotiations* [9] (where negotiations exert influences on one another) in a dynamic system that needs to generate solutions quickly to satisfy demand across a multi-echelon production scheduling network. Multi-linked negotiations occur in situations where a task requires further sub-tasks to be completed, and also when the existence of many such tasks results in competition for a common resource. An example of this is in a manufacturing supply chain network that consists of entity nodes and linkages defining contractor-contractee relationships. Very often such relationships do not extend beyond a direct relationship. Suppliers upstream usually have no information about their customer's customer, and the converse is true. Agent implementations usually simulate a single tier commodity market without such multi-tier relationships. Applying single-tier agent negotiation strategies to multi-tier systems brings up the issues of negotiation ordering and the parameters of the negotiations.

In this paper, we consider a dynamic multi-echelon production scheduling network involving the production, assembly and transportation of multi-indenture goods to satisfy dynamic requests. Such problems arise in an operational supply chain system, both commercial and military. A finished good undergoes component production and different levels of assembly at different facilities. Different facilities have different capabilities and capacities in providing the various operations required. A request is composed of a number of goods required at a number of locations by a certain time. It is handled by a management (contractor) agent who generates a schedule that optimizes the total production cost. The facilities and transportation services are represented by contractee agents who negotiate on the available jobs according to their local utilities. In Figure 1 for example, a request requires the following services to be provided: (1) production (or simply retrieval in our case) of items A and B, (2) transportation of items A and B to their next location, (3) assembly of items A and B to form item C, and (4) transportation of item C to its final destination.

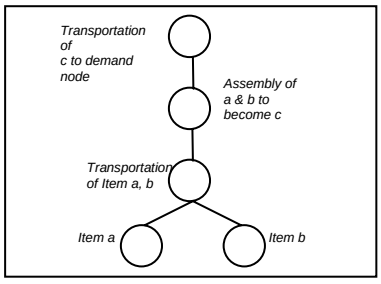

Figure 1: Sub-jobs required to fulfill a job

Contractor and contractee agents generally have no visibility over each other's interest. The challenge is to generate a schedule in a decentralized fashion that minimizes the (global) total production cost from the contractor's perspective while maximizing the contractee agents' local utilities. Hence our problem exhibits a hybrid flavor between a classical centralized scheduling problem (which minimizes metrics such as production cost) and a pure market-based resource allocation problem (which maximizes agent utilities) via approaches such as contract nets, auctions and negotiations. In this paper, we model the global production cost function and the local utility functions as economic functions. In both cases, the concept of diminishing marginal returns is applied to model the supply of production and transportation resources.

Due to the directly and indirectly linked relationships brought about by a multi-echelon network and multi-indenture goods, approaches based on single independent negotiations will not work well in general. To overcome this shortcoming, the concept of *negotiation ordering* and *feature assignment* was proposed in [9]. Our work differs from [9] in the problem solved: while [9] outputs a negotiation ordering and corresponding feature assignments for a subsequent negotiation phase, we assume that the actual duration of negotiations is negligible, hence enabling us to embed (integrate) the negotiations within our scheduling algorithm. As we will see later, this assumption is reasonable since agent negotiation is applied in our context as a computational and decision coordination step, which can be computed instantaneously. No latency is involved in decision making, assuming that the underlying agent communication network is reliable and fast. Our integrated approach allows us to do away with the uncertainties of negotiation outcomes and generate a final production schedule efficiently from the computational standpoint.

In section 2, we discuss related literature. In section 3, the problem is defined and an agent model presented. This is followed by our solution approach in Section 4. Experimental results on a real-world multi-echelon production scheduling network are presented in Section 5, followed by conclusions in section 6.

2. Related Works

[9] proposed a method for efficiently managing multi-linked negotiations, which occurs when one agent needs to negotiate with multiple agents about different issues which affect one another. Contrary to [9] where negotiation is time-consuming and the actual tasks execution is subject to the uncertainty of subsequent negotiation outcomes, in our context, negotiation can be efficiently carried out and hence embedded within the scheduling algorithm.

[6] provided an incentive based approach for agent-based production scheduling in a single-echelon system. A request is composed of a sequence of tasks to be completed, and each machine agent's capability may have different costs and speed for the same task. A market mechanism coupled with an incentive scheme drives the system towards a preferred state in which system wide scheduling constraints are met. The negotiation is based on a contract-net protocol, where the weakness is that earlier tasks are allocated first, thereby constraining resources for successive tasks.

[1] employed a 2 stage approach called MACOA (Multi-Agent Monte Carlo Configuration and Optimization Approach) to solve a dynamic production schedule problem. In the first stage, agents bid for jobs pulled from a job sequence in a way similar to [6], but includes the use of a common Transporter agent. In this first stage, jobs are also bidded upon sequentially starting from the earliest predecessor task in a task sequence. The paper, however, does not provide for considerations in assignment to parameters for the multi-linked negotiations e.g. allowed start time, deadline, reward of a job, etc. Similarly, the constraints imposed by earlier successful negotiations on subsequent ones were also not addressed.

[10] proposed tackling negotiations on 2 different levels. The upper level deals with the formation of high level goals and objectives for the agent, and decides whether or not certain agents should be involved in negotiations to reach certain objectives. The lower level deals with the feasibility and implementation operations, and involves the refinement of rough commitments formed at the upper level.

[2] described a holonic agent system framework implemented in JADE to cope with distributed manufacturing planning. In the same fashion, an earlier work in [5] proposed a mechanism to overcome infeasibility in negotiations due to directly-linked relationships. A task manager was to generate task holons for the bidding of resource agents in charge of many types of resources. A job is a list of operations that are negotiated on by the agents. The "indecision problem" is considered where a common resource is required for 2 or more jobs that require them within the same time. To prevent this conflict of time and resource, the task manager does not release tasks based on their arrival, but on their assigned priority, at the same time checking for overlap of both resource and time. The task holons can decide if a particular resource is no longer important should it be blocking the successful negotiation of the task. Control is then exercised over the tasks and resources by the task manager to release the go-ahead for the negotiation based primarily on the decision of holons.

The control criteria that define an agent's decision making were classified into 2 categories in [7]; *structural information* affects the scope of the agent control process, while *value information* affects the actual interaction and reasoning in complex agent relationships. Structural information can be the strategies that an agent may take in a given situation, or the decision of which other agents it may interact with. Value information pertains to quantifying the interactions or strategies using, say, a utility function, such that an evaluation can be made. To compare between disparate local utility functions, additional functions that acts very much like a currency exchange were defined. In this paper, we normalized these values into a single currency that can be used globally without conversion functions.

Earlier works on agent negotiations were mostly on single product market economy systems [3, 4, 8].

Market forces are used to allocate resources based on the prices set by supply and demand. For example, [4] is concerned with bidding for the right to provide electricity in the power market. Since these negotiations are single-tier, there were no issues of dependencies such as indirectly or directly-linked relationships.

3. Problem Formulation and Modeling

In this section we define the problem formally and present a multi-agent model for this problem.

3.1. Problem Definition

We define a *request* as a list of finished goods required at specified locations by a stipulated deadline. This list of finished goods requires a hierarchical level of assembly from components at different locations, and transportation between these locations. A request is hence a hierarchy of *m* jobs and sub-jobs which are either production (i.e. assembly) or transportation operations.

Consider the case in Figure 2 with a request that requires jobs 1 and 2 to be completed by a certain deadline at a given demand point. Both jobs 1 and 2 form disjoint trees representing the decomposition hierarchy of sub-jobs. A typical job hierarchy is shown in Figure 1. For simplicity, where there is no ambiguity, we shall collectively refer to jobs and sub-jobs as jobs for the rest for the paper.

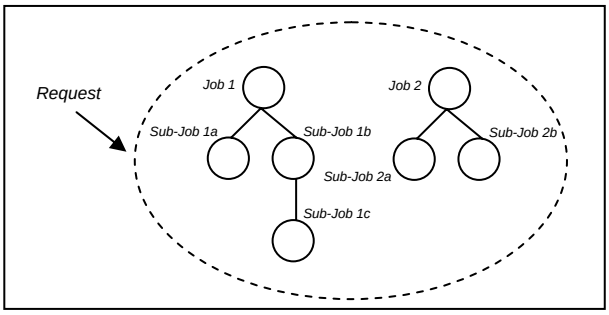

Figure 2: Multiple jobs in a single request

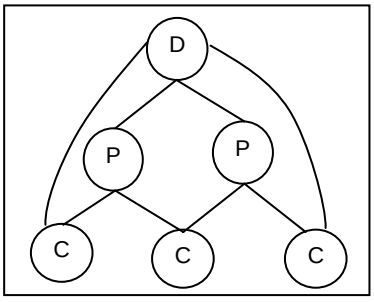

Figure 3: Network topology

Executing a job entails consuming a certain amount of renewable resources (such as manpower and equipment) at a single facility for a given duration. Different types of facilities may support different types of jobs. Transportation jobs require the use of transporters and incur traveling time between facilities.

Figure 3 shows a typical multi-echelon supply chain network topology where nodes represent facilities and directed edges represent the transportation links between the facilities. *D* and *P* (Distribution center and Production facility) are 2 types of facilities that store/supply goods and components and perform assembly jobs. Each facility has a maximum capacity for labor and inventory storage. *C* (Customer) represents the demand nodes.

Hence, there are 3 types of constraints in this problem: (a) deadlines of requests, (b) finite resource capacity at each facility, and (c) finite number of transporters.

The problem is to generate/revise a *schedule* as each request enters the system, which consists of:

1. the facility where each job is executed (also known as a *facility assignment*)
2. starting and ending times of the jobs at each facility
3. transport assignment to deliveries, *ETD* and *ETA* at pickup and drop locations.

The objective function is two-fold, in addition to the classical objective of production cost (which can be instantiated as makespan, resource cost, and the likes). Agents also possess *local utilities* that model foreknowledge, preferences, self-interest. Note that in our problem, the local utilities are *dynamic* in the sense that they are dependent on the agent's resource utilization profile at the time a job is to be assigned to it.

If we were to solve the above problem in a centralized fashion, it might be mapped into a classical bi-criteria job-shop scheduling problem which is well-documented in the literature. Unfortunately, in our context, local utilities may *not* be communicable to a central server due to privacy, security or operational limitations (such as network failures). Modeling the aspect of decentralized foreknowledge becomes necessary in a real-time operational context where requests arrive dynamically and are fulfilled one after another. Where no decommitment is allowed once a job is scheduled, it is necessary to rely on the collective wisdom (foreknowledge) of all agents within the supply chain to ensure that requests continue to be satisfied over time. Other than foreknowledge, the objective is hence to seek a balance between global cost minimization and local utility maximization through the process of negotiation.

3.2. Agent Model

Our model comprises 2 different types of agents: the management agent (*ma*) and *n* facility agents (*fa*). The *ma* receives the request and seeks a schedule that minimizes the sum of production costs P_j for all jobs. Each *fa* maintains its own local schedule of jobs

assigned so far and wishes to maximize its local utility function U_i. Hence, our goal is to find a schedule χ that minimizes $Z(\chi) = \sum_{j=1}^{m} P_j(\chi) - \sum_{i=1}^{n} U_i(\chi)$.

These two functions may be defined depending on the context of the problem, and may conflict with each other. In our case, to describe the economic effects of decreasing marginal returns, we model the unit production cost function p_j and unit utility function u_i as non-linear functions with decreasing rates of increase. Details of these functions will be discussed in the next section.

We view our problem as a multi-linked negotiation problem [9], where the *ma* negotiates with multiple *fas* about different issues which exert influence over one another. For example, in Figure 2, the negotiation for the completion time of job 2*a* affects job 2 in what is termed a *directly-linked* relationship as defined in [9]. Suppose negotiation of job 2*a* results in a completion time of 40, the negotiation of job 2 is constrained to having a start time of at least 40. Negotiations also influence each other in an *indirectly-linked* relationship. In Figure 2, suppose job 1 and job 2 are performed at the same facility, then it may be possible that they utilize the same resources should their timings overlap. *Indirectly-linked* relationship exists when negotiations compete for a common resource at the same facility.

One may perform multi-linked negotiations based on concurrent or sequential negotiations. Concurrent negotiations, where all possible negotiations are considered at the same time, often lead to an infeasible production schedule as the negotiations are done independently with no considerations on the mutual impact. More negotiation rounds may be required before the solution obtained is feasible, although the solution may also subsequently be more globally optimal. On the other hand, a sequential negotiation approach constrains later negotiations based on earlier successful negotiations. The greater flexibility enjoyed by earlier negotiations may be selfishly utilized by the agents involved such that resources or time become unavailable for later negotiations, hence resulting in overall failed negotiations. Without considering the *order* of negotiations, earlier negotiations experience a higher probability of success while later ones experience increasingly lower success probability.

In [9], the negotiations are considered in real time, and at the time of solving, there is no certainty to the outcomes (success or failure) of the negotiations. Hence, given n negotiations, it was required to enumerate and evaluate all the possible 2^n outcomes for a subsequent negotiation phase. In our work, we assume that the actual duration of negotiations is negligible with respect to the time horizon over which a solution is required. This allows us to embed the negotiations within the scheduling algorithm. In effect, our integrated approach enables us to do away with the uncertainties of negotiation outcomes and generate a final production schedule efficiently.

We propose a negotiation scheme where a series of negotiations will proceed sequentially in a given order (to be explained below), each involving the *ma* with a particular *fa* on a particular job, and the outcome of an early negotiation will become a constraint for later negotiations. The challenge is hence to find a negotiation ordering and a facility assignment such that negotiations proceeding along that ordering will produce a schedule that minimizes *Z*. All *fa*s are assumed to be truth-telling and non-collusive.

4. Three-Phase Solution Approach

Our algorithm is an iterative local search comprising of 3 sub-phases as shown in Figure 4: (a) find a facility assignment, (b) find a negotiation ordering, and (c) generate the best schedule that minimizes *Z*.

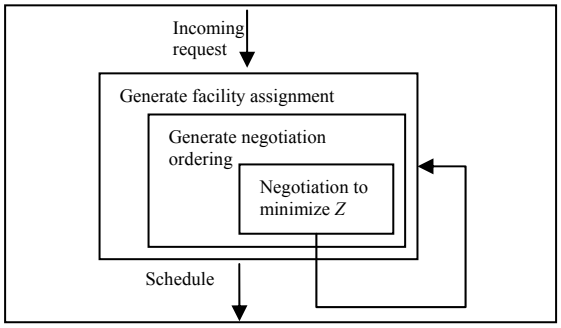

Figure 4: Solution approach

The notations in our algorithm are given as follows.

φ	Assignment of facility *i* to job *j*
ϕ	Negotiation ordering
χ_P	*ma*-generated schedule based on minimizing production cost
χ_C	*fa*-counter proposal schedule based on maximizing its own utility
v_j	Negotiation between *ma* and *fa* over the assigned job *j*, attributes defined below
M_i	Resource capacity of facility *i*
$v_j.st$, $v_j.dl$	(input) Allowed time window (start time and deadline respectively) between which job *j* must be completed
$v_j.h$	(input) Resource requirement of job *j*
$v_j.d$	(input) Duration of job *j*
$v_j.start$, $v_j.finish$	(Output) Actual start / finish time of job *j*. $v_j.finish - v_j.start = v_j.d$ $v_j.start$ is negotiated between the *ma* and *fa*.

Figure 5 shows a facility's local schedule represented by a resource utilization profile over time, which gives an illustration of the above notations:

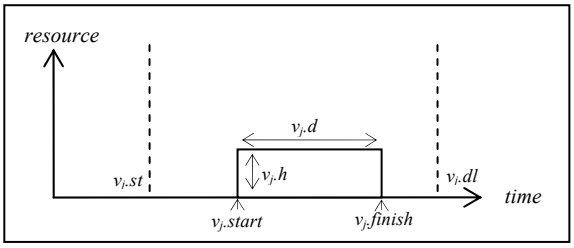

Figure 5: Facility local schedule

Next, we discuss the functions describing the production cost and local utility functions. For this purpose, consider a job *j* that is to be assigned to a particular facility at the time point $v_j.start$. Two parameters *u* and *x* are defined as follows.

u Worst-case utilization (i.e. fraction of utilized capacity between 0 and 1) at the assigned facility resulting from this assignment. For example in Figure 5 for job *j*, utilization is computed as $v_j.h/M_i$. If more than one jobs are executed at any time *t*, then the numerator will be the total resources utilized by these jobs at time *t*. *u* is the maximum utilization across all time points.

x Earliness of job *j*, between 0 and 1, defined as $\frac{v_j.d + v_j.dl - v_j.finish}{v_j.dl - v_j.st}$.

The unit production cost of a job *j* associated with this assignment is a function of *u*, denoted $p_j(u)$ (whose actual definition is an input to the problem, one instance of such is presented in our experiments in Section 5). The production cost of job *j* is hence given by $P_j(u) = p_j(u) \times v_j.d \times v_j.h$.

An agent *i*'s utility function associated with job *j*'s assignment is a function of *u* and *x*, denoted $U_i(u\ x)$[1].

Note that the functions p_j and U_i are measures of motivational quantities and are context-dependent. Although we used *u* and *x* as arguments for these functions, other arguments may also be used instead.

4.1. Phase 1: Find Facility Assignment φ

This phase is concerned with allocating a *fa* to each job. Here we apply a simple hill-climbing local search algorithm to generate different facility assignments from one iteration to the next. The initial facility assignment is generated greedily based on capacity, utilization, and transportation distance. Note that at this point, the actual schedule of the assigned jobs has not yet been determined, until Phase 3 (Section 4.3). For each assignment, a classical project scheduling algorithm that considers only the total production cost is then applied to determine an optimized schedule χ_P which consists of the start time $v_j.st$ and deadline $v_j.dl$ for each job *j* in the request.

4.2. Phase 2: Find Negotiation Ordering ϕ

Given a fixed facility assignment φ, this phase finds the best negotiation ordering. The initial negotiation ordering is generated in a bottom-up lexicographical manner of the job hierarchy (which follows a conventional way of how a multi-linked negotiation can be carried out sequentially). Using simulated annealing, successive negotiation orderings, consisting of sequenced facility-job pairings from Phase 1, are generated. Each (φ, ϕ, χ_P) triplet is the input for the next phase.

4.3. Phase 3: Agent Negotiation

For each triplet (φ, ϕ, χ_P), the negotiations between *ma* and the respective *fa*s in φ proceeds in the order defined in ϕ.

Consider an arbitrary job *j*. The *ma* first computes the optimal cost schedule χ_P for this job with respect to the unit production cost function $p_j(u)$ (using any classical algorithm in job shop scheduling). The corresponding starting time of job *j* $v_j.start$ is then proposed to the assigned *fa* based on a minimizing the value of $p_j(u)$. The *fa* then evaluates this based on the utility function $u_i(u, x)$. The *fa* may then counter-propose an alternative schedule χ_C that maximizes its $u_i(u, x)$. To determine if the *fa* takes this decision, we propose a reward scheme where units of credits flow among *fa*s through the *ma*. Initially, the *ma* possesses all credits. The local schedule χ_C and current credit standing will be used to negotiate with the *ma* as follows:

i. *fa* accepts χ_P unconditionally, if $U_i(\chi_P) \geq U_i(\chi_C)$
ii. *fa* counter-proposes χ_C and gives up $U_i(\chi_C)$-$U_i(\chi_P)$ credit units, if $U_i(\chi_P) < U_i(\chi_C)$ and *fa* has sufficient credits
iii. *fa* accepts χ_P with an increase of $U_i(\chi_C)$-$U_i(\chi_P)$ units credited, if $U_i(\chi_P) < U_i(\chi_C)$ and *fa* has insufficient credits

In case of ii, the *ma* will accede if through accommodating χ_C, it is able to generate a feasible schedule whose total production cost is increased by an amount no more than $U_i(\chi_C)$-$U_i(\chi_P)$; or in case of iii, if it has enough credits. Otherwise, the negotiation is unsuccessful. The result of a successful negotiation will in turn become a constraint for the subsequent negotiations in the ordering. A new schedule is

[1] In [9], the parameter *flexibility* was defined to measure the ratio of a job duration to its given time window. This parameter was used to determine an agent's probability of success in a negotiation, thereby making *flexibility* an indirect argument for an agent's utility. In this paper, we extend our agent's preference to have a job finished as early as its utility function decides (hence earliness *x*), and as balanced by its utilization *u*.

formed when all negotiations are successful. The fitness of this schedule χ is measured with the objective function Z. Our algorithm seeks to find the triplet (φ, ϕ, χ) that minimizes Z.

This proposed credit system acts as a control that prevents the *fa*s from making runaway changes to the *ma*'s schedule. Without this system in place, it is likely that the final schedule foregoes too much on cost minimization for the sake of maximizing utility. Having this credit system therefore enforces a balance between the global and local agent objectives.

4.4. Feedback to Phase 2 and Phase 1

The negotiation activities in phase 3 will serve as a feedback to generate next moves in phases 1 and 2, described as follows.

To generate a new negotiation ordering in phase 2, a k-OPT heuristic is applied, as follows:

1. Sort the jobs associated with the successful counter-proposals in ascending order of the increase in utility $U_i(\chi_C)$-$U_i(\chi_P)$ and select the top k of them. Let this list be denoted $x_1,\ldots,x_k$.
2. Sort the jobs associated with the unsuccessful counter-proposals in descending order of the increase in utility and select the top k of them. Let this list be denoted $y_1,\ldots,y_k$.
3. The new ordering is obtained by swapping the job x_i with y_i in the current ordering, for i=1 to k.

In essence, this heuristic allows the poorer but successful counter-proposals to make way for the better but unsuccessful counter-proposals. By swapping their positions in the ordering, we allow the unsuccessful but potentially higher utility counter-proposals a better chance of acceptance, since it is placed in a position for which a poorer counter-proposal was previously successful.

For the feedback to phase 1, we apply the following heuristic:

1. For facilities of the same type, sort in descending order of the value τ_i, which is defined as the sum of all the jobs' contribution for facility i to Z divided by the total resource-hours consumed by these jobs. Intuitively, a lower τ_i value indicates a facility that is more resource-efficient.
2. Select the facility with the highest τ_i and select the job with the highest production cost from this facility.
3. Fit this job into the facility with the lowest τ_i if the increase in τ_i in this target facility is less than the decrease in the source facility. Otherwise, generate a random move.

This heuristic attempts to generate a move that minimizes Z.

5. Experimental Results

In our experiments, we like to investigate the effectiveness of our negotiation protocol in incorporating agent utilities. More precisely, we study the behavior of the Z value and what-we-term the δ *value*, which is the marginal relative gain in utility value per unit relative increase in production cost due to negotiation. We compare the values of Z and δ under the cases with and without negotiation.

In both cases, we model the unit production cost function $p_j(u)$ as follows (as shown in Fig 6):

$p_j(u) = A + B\tanh^{-1}(\alpha u)$, where the values of A, B and α are constant and are set to 4, 1 and 1 respectively for all test sets. We choose this function to model the behavior of convex production cost that tends to ∞ as utilization approaches 1.

For the former case, phase 3 involves the *ma* computing the optimal solution that minimizes the production cost without entering into negotiation with the *fa*s. The Z value of this optimal solution is recorded.

For the latter case, the full-fledged phase 3 and feedback algorithm discussed in Sections 4.3 and 4.4 is executed to derive the Z value. For this purpose, we represent the facility's unit utility function $U_i(u, x)$ as follows (as shown in Fig 7):

$$U_i(u,x) = C - Dx\ln(\frac{\beta u}{x} + \varepsilon).$$

We choose this function to model the economics of diminishing marginal utilities. To simulate different behaviors in the facilities with respect to the utility function, 3 sets of parameter with varying values of C, D, β and ε are used, as shown in Table 1.

	fa Behavior	Parameters
Set 1	All *fa*s prefer earliness Little importance on reducing utilization	$2.0 \le C \le 4.0$ $0.5 \le D \le 1.0$ $0.05 \le \beta \le 0.10$ $\varepsilon = 0.001$
Set 2	All *fa*s prefer earliness, High importance on reducing utilization	$3.0 \le C \le 5.0$ $0.5 \le D \le 1.0$ $1.75 \le \beta \le 2.5$ $\varepsilon = 0.01$
Set 3	Equal mix of Set 1 and Set 2 behaviors	

Table 1: *fa* behaviors parameter sets

For the feedback heuristics, we experimented with k=1 (i.e. we implemented a basic swap neighborhood).

Due to the probabilistic nature of phases 1 and 2, we used the same random seed for both cases to force the local search to behave identically. In doing so, the different solutions derived are a result of negotiation and feedback rather than the underlying local search.

The request sets for the experiments were generated with loose and tight planning horizons as shown in Table 2.

Request Set	Total No.of Jobs	Planning Horizon
R1a	70	500
R1b	70	2000
R2a	350	500
R2b	350	2000
R3a	750	500
R3b	750	2000

Table 2: Request data sets

The results of our experiments are shown in Tables 3, 4 and 5. For each test set, the value δ is computed by taking the ratio of relative utility gain to relative production cost loss, e.g. in Table 3 R1A, 28.82 = (61759-57194)/57194÷(37290-37187)/37187.

Set 1	Neg	Total Utility	Total Production Cost	Z	δ
R1A	No	57194	37187	-20007	28.82
	Yes	61759	37290	-24469	
R1B	No	55297	37098	-18199	156.09
	Yes	62975	37131	-25844	
R2A	No	379137	270654	-108483	2.30
	Yes	385624	272670	-112954	
R2B	No	398674	264142	-134532	11.48
	Yes	409733	264780	-144953	
R3A	No	611669	461661	-150008	1.12
	Yes	624588	470360	-154288	
R3B	No	653171	442485	-210686	6.87
	Yes	670635	444207	-226428	

Table 3: Results for Set 1

Set 2	Neg	Total Utility	Total Production Cost	Z	δ
R1A	No	55136	37187	-17949	7.00
	Yes	55655	37237	-18418	
R1B	No	58367	37098	-21269	33.86
	Yes	59699	37123	-22576	
R2A	No	319365	270654	-48711	2.44
	Yes	323258	272005	-51253	
R2B	No	368432	264142	-104290	5.78
	Yes	371246	264491	-106755	
R3A	No	490252	461661	-28591	2.53
	Yes	503158	466469	-36689	
R3B	No	575549	442485	-133064	8.39
	Yes	585966	443440	-142526	

Table 4: Results for Set 2

Set 3	Neg	Total Utility	Total Production Cost	Z	δ
R1A	No	57009	37187	-19822	9.19
	Yes	57742	37239	-20503	
R1B	No	59885	37098	-22787	2.99
	Yes	60030	37128	-22902	
R2A	No	359856	270654	-89202	2.18
	Yes	364884	272385	-92499	
R2B	No	390158	264142	-126016	8.27
	Yes	396863	264691	-132172	
R3A	No	572144	461661	-110483	1.19
	Yes	580301	467209	-113092	
R3B	No	630463	442485	-187978	5.41
	Yes	639912	443711	-196201	

Table 5: Results for Set 3

It can be seen that, as the *ma* in the first 2 phases does not account for each *fa*'s preference for earliness, all the runs with the negotiation phase (phase 3) consistently gives a better objective value Z than without negotiation. This improvement ranged from 1% to 42% depending on the quality of the initial facility assignment. Furthermore, the marginal values gained from negotiation are quite large, suggesting that our proposed approach is effective in deriving a schedule that gains agent utility at a small expense of loss in total production cost.

In modeling the local preference of *fa*s as defined, we have taken into account a local aspect of information that is not visible to the *ma*'s perspective of achieving a schedule with minimum production cost. It is worth noting that with our local utility functions, the algorithm returns consistently lower production cost on the negotiated schedule in Set 2 than in Set 1. The parameters with which we used to tune the behavior of the facility agents performed as expected, with the Set 2 *fa* being more cost aware (and with less preference for earliness) relative to Set 1 *fa*s. A study of the utility functions $u_i(u,x)$ would reveal that as the parameters β and ε increase, the desirability of earliness over minimizing utilization becomes increasingly less. Similarly, Set 3 *fa*s, which are a mixture of behaviors in Sets 1 and 2, generated production costs that lie mostly between those of Sets 1 and 2.

The experiments were conducted on a Pentium-IV workstation running at 2.4GHz clock speed with 512Mb of memory. Agent communications were simulated by a sequential tread so that communication overheads become negligible. Phase 1 was reiterated 50 times, while Phase 2 was reiterated 20 times. The smallest test cases (R1a, R1b) took less than 5 seconds of runtime, while the largest (R3a, R3b) incurred about 5 minutes. These demonstrate the computational efficiency of our approach.

6. Concluding Remarks

In the context of a dynamic and secure networked system, it may be impossible to make centralized decisions based on complete information. The use of a decentralized algorithm in solving complex multi-echelon scheduling problems is not only essential, but also more efficient computationally.

Moreover, the use of decentralized decision making through multiple agents enables uncertainty of future demands to be tackled through collective wisdom among agents. By distributing the different foreknowledge to their respective owners (i.e. having the *ma* only deal with global production cost while the *fa*s counter-balancing with their foreknowledge), our approach allows one to obtain a production schedule that effectively hedges against uncertainty.

We saw the complexity of multi-linked relationships that are prevalent in today's supply chain networks. Inspired by the work of [9] on negotiation ordering in a multi-linked system, we were led to design an efficient local search algorithm centering on multi-linked negotiation ordering. This approach has led us to solve the multi-echelon multi-indenture production scheduling problem effectively in a distributed fashion that avoid the pitfalls of sequential or concurrent negotiations.

Our current approach processes the requests on a first-come-first-served basis, since there is no visibility on future orders, as inherent in dynamic systems. If the requests were to follow a prescribed demand distribution, we think it would be interesting to investigate the relationship between demand patterns and agent utility functions. On another front, we think it is interesting to investigate the effect of different credit systems on the results. For instance, one might consider the setting where both *ma* and *fa*s possess an initial non-zero amount of credits before the beginning of negotiation.

References

[1] W. T. Goh and Z. Zhang, Multi-agent system for dynamic production scheduling and optimization, In Proc. 3rd *International Symposium on Multi-Agent Systems, Large Complex Systems, and E-Businesses*, 664-674, Oct 2002

[2] D. Kotak, S. Wu, M. Fleetwood, and H. Tamoto. Agent-based holonic design and operations environment for distributed manufacturing, *Computers in Industry*, Volume 52, 95-108, 2003

[3] V. Krishna, V.C. Ramesh; Intelligent agents for negotiations in market games, part 1: Model; *IEEE Transactions on Power System* Volume 13, No. 3, 1103-1108, Aug 1998.

[4] V. Krishna and V.C. Ramesh. Intelligent agents for negotiations in market games, part 2: Application; *IEEE Transactions on Power System* Volume 13, No. 3, 1109-1114, Aug 1998.

[5] P. Sousa and C. Ramos. A distributed architecture and negotiation protocol for scheduling in manufacturing systems; *Computers in Industry* Volume 38, 103-113, 1999.

[6] J. Vancza and A. Marcus. An agent model for incentive-based production scheduling. *Computers in Industry*, Volume 43, 173-187, 2000

[7] T. Wagner and V. Lesser; Relating quantified motivations for organizationally situated agents; *Intelligent Agents VI -Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence,* Volume 1757, 334-348, 1999.

[8] M.P. Wellman and P.R. Wurman. Market-aware agents for a multiagent world; *Robotics and Autonomous Systems* Volume 24, 115-125, 1998.

[9] X. Zhang, V. Lesser, and S. Abdallah. Efficient Management of Multi-Linked Negotiation Based on a Formalized Model. *Autonomous Agents and Multi-Agent Systems*, Volume 10, Number 2, 165-205. 2005.

[10] X.Q. Zhang, V. Lesser, and T. Wagner. A Layered Approach to Complex Negotiations. *Web Intelligence and Agents Systems: An Int'l Journal,* Volume 2, Number 2, 91-104, 2004

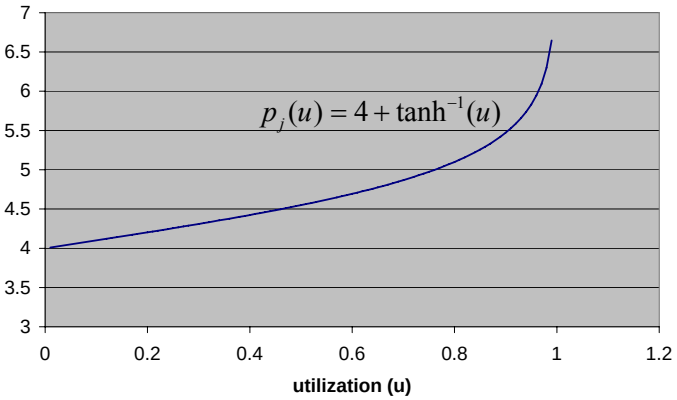

Figure 6: Example of Unit Production Function

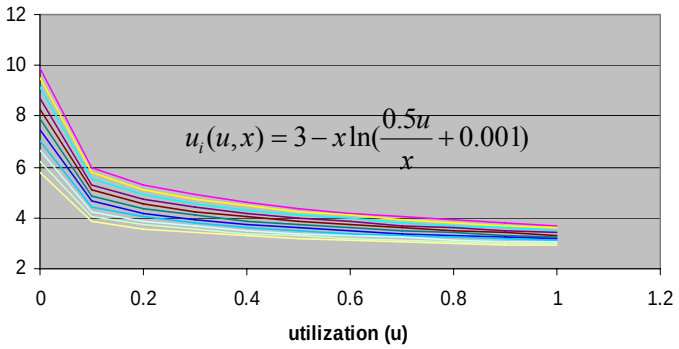

Figure 7: Utility Functions for different values of *x*