

Solving the n -job 3-stage flexible flowshop scheduling problem using an agent-based approach

ASTGHIK BABAYAN[†] and DAVID HE^{†*}

In this paper, a general methodology of agent-based manufacturing systems scheduling, incorporating game theoretic analysis of agent cooperation is presented to solve the n -job 3-stage flexible flowshop scheduling problem. The flowshops are flexible in the sense that a job can be processed by any of the identical machines at each stage. Our objective is to schedule a set of n jobs so as to minimize the makespan. We perform error bound analysis using the lower bound estimates developed in the literature as a datum for comparing the agent-based scheduling solutions with other heuristic solutions. The results of the evaluation show that the agent-based scheduling approach outperforms existing heuristics for the majority of the testing problems.

1. Introduction

Because of their dynamic nature and their practical interest in industrial applications, manufacturing scheduling problems have been the subject of extensive research in the past few decades. Scheduling problems are generally NP-hard, i.e. it is unlikely to find an optimal solution without the use of an essentially enumerative algorithm and the computational time increases exponentially with problem size. In a manufacturing system, the scheduling problem is further complicated when unforeseen events occur in the system. Agent-based technology paves a new way of thinking about many complex systems and overcoming the complexity problem. Traditionally centralized manufacturing planning, scheduling, and control mechanisms are being found to be insufficiently flexible to respond to changing production operations and highly dynamic variations in market requirements. The centralized organizations may also result in much of the system being shut down by a single point of failure, as well as plan fragility and increased response overheads.

In this paper, an agent-based manufacturing scheduling methodology is applied to solve the n -job 3-stage flexible flowshop scheduling problem. The n -job 3-stage flexible flowshop scheduling problem has been solved in the literature (see Soewandi and Elmaghraby 2001). However, the solutions developed in the past are problem dependent, i.e. they are developed for solving a specific scheduling problem. The agent-based scheduling methodology presented in this paper is a general one and is suitable for solving large-scale scheduling problems where the overall scheduling task can be categorized into three sets: schedulable jobs, non-schedulable jobs and

Revision received June 2003.

[†]Department of Mechanical and Industrial Engineering, The University of Illinois-Chicago, Chicago, IL 60607, USA.

*To whom the correspondence should be addressed. e-mail: davidhe@uic.edu

scheduled jobs, at any particular stage of the scheduling process. The presented methodology does not impose any limitations on manufacturing system structures.

The key feature associated with agent-based scheduling is the cooperation and coordination of the agents. For coordination of the agents, classical tools developed in games theory are applied for solving the problem. In particular, the cooperative model of games is considered, where agents form coalitions to maximize the payoff from their cooperation. The power index of an agent corresponds to the Shapley (1953) value for that agent associated with the scheduled completion time in the system achieved by the agent. The overall scheduling process is an integration of defined inner and outer games where the outer game is among the agents eligible to schedule their jobs in the system and the inner game is among the agents eligible to *re-schedule* their jobs in the system.

The purpose of this paper is to introduce the agent-based scheduling for solving hard scheduling problems and to show its application to n -job 3-stage flexible flowshop scheduling problems. The flowshops are flexible in the sense that a job can be processed by any of the identical machines at each stage. Our objective is to schedule a set of n -jobs so as to minimize makespan. We perform error bound analysis using the lower bound estimates developed in the literature as a datum for evaluation of agent-based scheduling solutions. The results obtained by our agent-based approach are compared with those by Soewandi and Elmaghraby (2001) who intensively examined the problem and developed better performing heuristic algorithms.

The remainder of the paper is organized as follows. Section 2 describes the scheduling problem. Section 3 summarizes the literature on developed scheduling methods, agent applications, and interaction among agents. The framework of the agent-based scheduling methodology is presented in section 4. Computational results from comparison of the agent-based scheduling method with existing methods are presented in section 5. Finally, section 6 summarizes the research and concludes the paper.

2. The scheduling problem

In the n -job 3-stage flexible flowshop scheduling problem, there are m_i identical machines at stage i , where $i = 1, 2, 3$ and a set of n simultaneously available jobs to be processed sequentially on three stages of a production facility. The jobs arrive at stage 1 where the corresponding operations are performed and the jobs are delivered to stage 2 for the completion of succeeding operations. Similarly, after completing operations at stage 2 the jobs are delivered to stage 3 for the final operations. The jobs are processed without pre-emption and there is unlimited buffer space available for waiting jobs between the production stages. The processing times of operations are known deterministically. The objective of scheduling is to assign jobs to the machines at the corresponding stages and determine the processing sequences on the machines so that the makespan (C_{\max}), i.e. the maximum completion time, is minimized. Graphically, the manufacturing system can be represented as in figure 1.

If we use a node to represent an operation and an arc as the precedence relationship, then a job i can be represented by a digraph, where O_{ik} , $i = 1, 2, \dots, n$; $k = 1, 2, 3$ corresponds to operation k of job i that should be performed on one of the machines at stage k . Note that the whole set of n jobs can be represented by a superimposed digraph. The superimposed digraph can be generated by connecting the final nodes (operations) of the jobs to a dummy node A_d with processing time equal to zero as shown in figure 2. This representation of the overall scheduling task helps to initialize

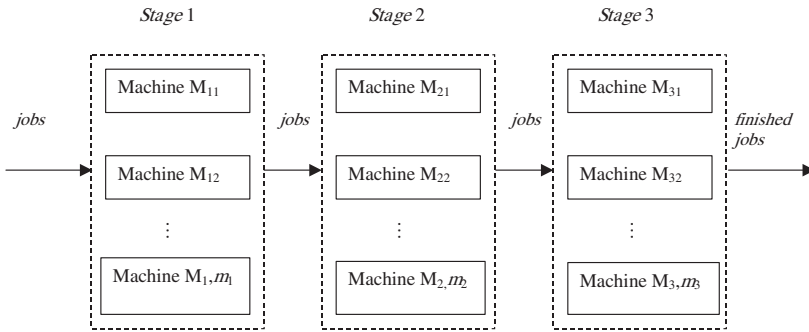


Figure 1. Structure of manufacturing system.

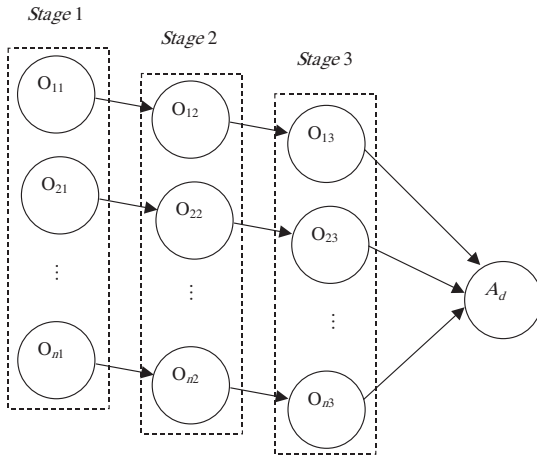


Figure 2. Representation of n jobs by a complex digraph.

the sets of schedulable jobs, sets of non-schedulable jobs, and the set of scheduled jobs at the initial stage of the scheduling process, and to update the sets later at every stage of the scheduling process. The details of the initialization process will be presented in section 4.

3. Literature review

Next, literature on solution methods developed for the scheduling problem, agent applications and cooperation mechanisms for agent coordination are summarized.

3.1. Related scheduling problems in literature

The n -job 3-stage flexible flowshop scheduling problem ($F3||C_{max}$) is shown by Garey *et al.* (1976) to be NP-hard in a strong sense. This implies that for $F3||C_{max}$ the existence of a polynomial time algorithm for finding an optimal solution is unlikely. Therefore, developing fast-heuristic/approximation algorithms yielding near-optimal solutions are of great interest. The flexible flowshop scheduling problem with more than two stages was first studied by Lee and Vairaktarakis (1994). They presented a heuristic for k -stage flexible flowshop scheduling and demonstrated that the heuristic has a *worst-case performance bound* (WCPB), which is computed as

a ratio of the makespan obtained by the heuristic, i.e. C_{\max}^H , over the least upper bound as presented in equation (1).

$$\frac{C_{\max}^H}{C_{\max}^*} \leq k - \sum_{i=2}^k \frac{1}{\max\{m_{i-1}, m_i\}}. \quad (1)$$

Guinet and Solomon (1996) developed several algorithms for minimizing maximum tardiness or maximum completion time of the n -job k -stage flexible flowshop scheduling problems. Their algorithms were derived based on the m -machine pure flowshop scheduling algorithms developed by Nawaz *et al.* (1983), Campbell *et al.* (1970), Townsend (1977), and priority-rule-based algorithms such as shortest processing time (SPT), longest processing time (LPS), etc. They also developed a lower bound and evaluated the performance of heuristics using the lower bound as the target values.

Soewandi and Elmaghraby (2001) studied the n -job 3-stage flexible flowshop scheduling problem and developed polynomial heuristic algorithms and compared the results with those obtained by the heuristic algorithm of Lee and Vairaktarakis (1994). Their heuristics were developed based on that of Lee and Vairaktarakis (1994) and they viewed the scheduling problem from the prospective of a flexible flowshop as a generalization of a regular k -stage flowshop.

The two heuristic procedures (SP1 and SP2) developed by Soewandi and Elmaghraby (2001) make use of a shop partitioning approach. They partitioned the 3-stage flexible flowshop problem into two subproblems: a two-machine flowshop problem, and a single-stage parallel machine scheduling problem, and demonstrated that SP1 and SP2 have WCPBs as presented in equations (2) and (3):

$$\frac{C_{\max}^{SP1}}{C_{\max}^*} \leq 4 - \frac{1}{\max\{m_1, m_2\}} - \frac{1}{m_3}, \quad (2)$$

$$\frac{C_{\max}^{SP2}}{C_{\max}^*} \leq \frac{10}{3} - \frac{1}{\max\{m_1, m_2\}} - \frac{1}{3m_3}. \quad (3)$$

In equations (2) and (3), C_{\max}^* corresponds to the optimal makespan, and C_{\max}^{SP1} and C_{\max}^{SP2} correspond to the makespans obtained by applying heuristic procedures SP1 and SP2, respectively.

These heuristics utilize the procedure H1 developed by Lee and Vairaktarakis (1994) and a new H2 heuristic procedure. The heuristic procedures H1 and H2 are based on finding Johnson's (1954) sequence for the first two stages and improving the solutions by applying different scheduling rules to stages, i.e. first available machine (FAM), last busy machine (LBM), and random decreasing machine (RDM).

The third heuristic, Three Stage-Critical Job (TS-CJ), was developed with an auxiliary-problem-based approach. At first, an initial sequence is obtained using a modified Johnson's algorithm based approach for the simple 3-stage flowshop problem with surrogate processing times developed by Chen *et al.* (1996) (CJ) or the heuristic developed by Nawaz *et al.* (1983) (NEH) with performance bounds as in equations (4) and (5).

$$\sqrt{3/2} \leq \frac{C_{\max}^{NEH}}{C_{\max}^*} \leq 2, \quad (4)$$

$$\frac{C_{\max}^{CJ}}{C_{\max}^*} \leq 5/3. \quad (5)$$

In equations (4) and (5), C_{\max}^{NEH} corresponds to the makespan obtained by applying heuristic procedure NEH and C_{\max}^{CJ} corresponds to the makespan obtained by applying heuristic procedure CJ.

In total, four different heuristic procedures were evaluated. They are summarized as follows:

- (1) SP1-H1: Sequence the first two stages using the heuristic of Lee and Vairaktarakis (1994) and then sequence the stage 3 tasks using RDM rule (this is similar to the FAM rule where the job to be assigned is selected randomly from the set of ready jobs).
- (2) SP2-H1: Sequence the first two stages as in SP1-H1, and then sequence stage 3 tasks following a modified Jackson's (MJ) procedure (Jackson, 1955). MJ mimics the least processing time (LPT) rule by taking into account the availability of the jobs from prior machines.
- (3) SP2-H2: Sequence the first two stages using procedure H2, then use the MJ procedure to solve the $Pm_3|r_j|C_{\max}$ (scheduling jobs with ready times, see Carlier, 1987) problem for stage 3.
- (4) TS-CJ: Implement procedure TS using procedure CJ for initialization.

Due to the symmetry of the problem, the roles of stages 1 and 3 were reversed and all heuristics were applied to the reversed problem. The obtained lowest makespan was considered as the best solution to the scheduling problem.

In their paper, Soewandi and Elmaghraby (2001) conducted extensive experiments to evaluate the performance of the developed heuristics. To obtain a datum for the comparison of all procedures, several lower bounds on the makespan of the original problem $F3(Pm_1, Pm_2, Pm_3) - |C_{\max}$ were developed. The best lower bound was taken to be equal to the maximum of all individual lower bounds. For a detailed discussion on the development of the lower bounds readers can refer to Soewandi and Elmaghraby (2001).

3.2. Agent-based applications

The notion of an agent is found in a wide range of research in computer science (CS), distributed artificial intelligence (DAI), etc. Agent-based approaches have been applied to problems with very large or unpredictable solution domains. The only reasonable way to address these problems is to develop a number of modular components that are specialized at solving a particular aspect of it. Researchers have applied an intelligent agent framework for modelling operational problems in the general manufacturing field. Real world examples of multi-agent applications include: power systems management (Jennings *et al.* 1995; Varga *et al.* 1994), particle accelerator control (Jennings *et al.* 1993), telecommunications network management (Weihmayer and Velthuisen 1994), spacecraft control (Schwuttke and Quan, 1993), computer integrated manufacturing (Parunak, 1995), process planning and scheduling for distributed virtual manufacturing (Wu *et al.* 2002), job shop scheduling (Liu and Sycara 1998; Morley and Schelberg 1993), project scheduling (Knotts *et al.* 2000), industrial warehousing (Byung-In *et al.* 2002), etc.

The Autonomous Agents at Rock Island Arsenal (AARIA) project funded by the US DARPA Agile Manufacturing and SBIR programs developed a factory scheduler based on autonomous agents that actively represent each step of manufacturing a part (see Parunak *et al.* 1997; Parunak *et al.* 2001). The project has demonstrated how agents can be placed on every productive resource, linked

through the Internet, and scheduled to run an effective virtual manufacturing enterprise that is self-configuring.

The multi-agent systems paradigm represents one of the most promising approaches to the development of agile scheduling systems in manufacturing (Rabelo *et al.* 1999). Scheduling problems have already attracted various efforts in multi-agent research (see for example Sousa and Ramos 1997; Rabelo and Camarinha-Matos 1994, etc). Agent technology was applied by Sycara *et al.* (1991a, 1991b) to develop distributed constrained heuristic search methods for solving a set of problems. In particular, a job shop scheduling problem was solved, where each agent is responsible for scheduling a set of orders. Bussmann and Schild (2001) present an agent-based approach for the control of flexible production systems. In their approach, agents represent tasks and resources, where resource agents bid in the auction of task agents. DaimlerChrysler performed a set of tests that demonstrated the feasibility and the benefits of the approach. A survey by Shen and Norrie (1999) reports 30 projects using agent technology for manufacturing planning, scheduling and execution control where agents represent physical entities, processes, operations, parts, etc. In general, distributed systems have the following advantages (Decker, 1987):

- (1) they can simplify the problem solving process by splitting the problem into simple tasks;
- (2) they can tolerate uncertain data and knowledge;
- (3) they offer conceptual clarity and simplicity of design;
- (4) they present graceful degradation in computational complexity;
- (5) they allow incremental modification of the system boundary; and
- (6) they are well suited to distributed problems.

The application of an agent-based approach to manufacturing scheduling is based on the idea that scheduling agility can be improved because of distributed autonomous systems and negotiation-based decision-making in a multi-agent environment.

Minsky (1986) suggested the use of a multi-agent system where there is a wide-range of reasonably self-contained pieces of functionality that are independently distributed in terms of timing resources.

A universal definition for '*agent*' or '*processor*' has not been found in the literature. However, it does not prevent people from expanding the application areas of agent technology. Sichman and Demazeau (1992) considered a '*processor*' as an agent when it possesses at least the following three properties:

- (1) it has a certain degree of autonomy to reason about and to make decisions by itself;
- (2) it has the capability to interact with other agents; and
- (3) it has the knowledge to solve independently a part of the global problem.

An agent can possess various properties depending on the nature of the problem under consideration and the environment. The agents involved in the decision making process do not necessarily possess the same functionality. The level of functionality is defined in the problem decomposition process. Some of the agent properties found in the literature (Genesereth and Ketchpel 1994, Galliers 1988, Rosenschein and Genesereth 1985, Wooldridge and Jennings, 1995) are appropriate

for the type of the agents used in our system. They are listed below along with the properties we believe our agents possess:

- autonomy* : agents operate without the direct intervention of human operators, and have high degree of computational capabilities;
- sociability* : agents interact with other agents via some kind of agent communication language;
- reactivity* : agents perceive their acting environment, and respond to the changes that occur there;
- pro-activity* : agents do not simply act in response to the environment, they are able to exhibit goal oriented behaviour by taking initiatives;
- veracity* : agent will not knowingly communicate false information;
- benevolence* : agents do not have conflicting goals, and every agent will therefore try to do what is asked of it; and
- rationality* : an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved.

3.3. Cooperation and negotiation of agents

In a multi-agent environment, the coordination among distributed parts is generally realized through negotiation and cooperation among agents. In the literature there are several models that use cooperation and coordination for agent interactions. Designing the agent coordination system is one of the key issues in agent-based system modelling and has a great impact on the problem's solution quality. The most popular mechanisms found in the literature include a voting mechanism (Fordyce and Sullivan 1994), contract net protocol (Smith 1980), and its modified versions such as extended contract net protocol by Fischer *et al.* (1995), the market driven contract net by Baker (1991), etc. Other examples of agent negotiation include bidding mechanism (i.e. Duffie and Piper 1986, Ow and Smith, 1988, Lin and Solberg 1992), resource oriented mechanism (Butler and Ohtsubo 1992, Shen and Norrie 1998), and others.

Smith and Davis (1981, 1983) discussed the suitable applications of contract nets as well as their limitations. The application areas include: (1) decomposable problems where the subtasks are large and they need intensive computational effort; (2) problems where the main objectives are reliability, bottleneck prevention, and distributing control; and (3) domains where the node task allocations are not known in advance. Despite some advantages of contract nets, i.e. simplicity, reliability, and scalability, they have their limitations such as: (1) determining the information that should be given with task announcements and bid-submission; (2) result sharing protocol; and (3) method of enhancing global system performance.

The other most popular agent coordination mechanisms are bidding mechanisms. Their features were studied by Ramamritham and Stankovic (1984). The authors concluded that bidding mechanisms are not very effective if all resources have similar capabilities and workload.

Agent interaction was coordinated by solution concepts developed in game theory. In the theory of games, solution concepts were developed to describe how the players/agents can distribute the payoffs of the cooperation among themselves given that the payoffs for all coalitions can be computed (see, for example Aumann 1959, Rapoport 2000). However, the game theory does not provide tools on how players can move from one coalition to another to improve their payoff. A

kernel-based coalition formation algorithm was developed by Shehory and Kraus (1999) for self-interested agents in non-superadditive environments that considers the cost associated with cooperation, distribution, and computation time.

The dynamic coalition formation algorithms were studied by Klusch and Gerber (2002) for open, distributed, and heterogeneous environments. They considered an environment, where each agent can freely enter and/or leave the society at any time. Cooperative models of games were considered for making coalitions and distributing payoffs among coalition members according to the coalition contracts. In particular, for low complexity computation of stable payoff distribution in superadditive environments, they suggested the bilateral Shapley value for payoff distribution.

Sandholm and Lesser (1997) addressed a coalition formation method that targets reducing the computational complexity in the number of agents. They studied coalition formation among the self-interested agents that need to solve combinatorial optimization problems assuming that the optimization problems can be solved exactly with lower costs.

As the number of agents increases, the number of possible coalition structures increases exponentially and the size of the problem of finding optimal coalition becomes too large. Sandholm *et al.* (1999) developed an algorithm that establishes a tight bound within this minimal amount of search, and they showed that any other algorithm would have to search more.

Kutanoglu and Wu (2002) modelled a resource scheduling problem as a schedule selection game in which the system proposes resource allocation for a particular period of time by posting a number of alternative resource schedules. Agents representing the jobs and competing for the resources have private information concerning the true benefits of each schedule to the job types they represent. In their schedule selection model, agent interaction is modelled as an N -person non-cooperative game with incomplete information, where each agent cares only about the profit received from the schedule.

In this paper, the cooperation and negotiation mechanism for an agent-based manufacturing system scheduling is developed based on cooperative game theory. Before the presentation of the cooperation and negotiation mechanism, the general scheduling framework under which the cooperation and negotiation mechanism works is first explained.

4. Agent-based scheduling framework

In this section, the agent-based scheduling methodology for solving the n -job 3-stage flexible flowshop scheduling problem is presented. Schematically, the general framework can be presented as in figure 3. The key elements of the agent-based scheduling framework include:

- (1) the methodology takes the manufacturing system structure and scheduling task structure as inputs and generates agents as entities in the system that are the ultimate participants in the scheduling process;
- (2) each agent is responsible for scheduling the corresponding job; and
- (3) the coordination among agents in the scheduling systems is carried out by solution concepts developed in cooperative game theory.

The key elements in implementing the agent-based scheduling framework are the development of methods for (i) problem decomposition and identification of agents;

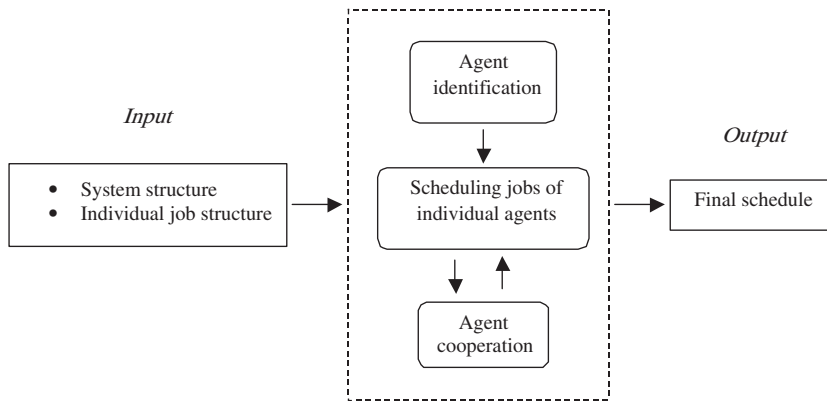


Figure 3. Agent-based scheduling framework.

(ii) scheduling of subproblems; and (iii) agent cooperation. Next, the details of these key developments are presented.

4.1. Problem decomposition and identification of agents

There are some characteristics that a system/problem should possess in order to be suitable for agent-based applications:

- (1) the system/problem should be decomposable, here the physical meaning of decomposition is not enough, therefore;
- (2) the system/problem should not lose its integrity because of decomposition; and
- (3) the independent subsystems/subproblems can be clearly distinguished.

To identify the agents in the system, we first need to decompose the scheduling problem into subproblems. The decomposition of the scheduling problem is executed by decomposing the superimposed digraph of the overall scheduling task into sub-digraphs. When the scheduling task is represented by a digraph, the precedence relationship between the jobs/operations can be used to identify the following sets of jobs: (i) jobs that are ready to be scheduled; (ii) jobs that cannot be scheduled because some or all of their preceding jobs have not been scheduled yet; and (iii) jobs that have been already scheduled in the system. For the scheduling problem described in section 2, the digraph representation is generated ‘artificially’ and agent identification could be achieved in a simpler way. However, we think it is worth discussing this for more general scheduling applications. In the context of scheduling, the three sets of jobs are defined as follows:

- SAJ* set of schedulable jobs. These are jobs that can be scheduled.
NSA set of non-schedulable jobs. These are jobs that cannot be scheduled.
SJ set of scheduled jobs. These are the jobs that have already been scheduled.
SJ is always empty before scheduling starts.

To each subproblem/job generated during the decomposition process an agent is assigned as an entity of the system that is responsible and authorized only for scheduling his/her job in the system. This way of decomposition preserves the integrity of the problem and offers flexibility of designing an agent-based scheduling system.

To apply the agent-based approach for solving an n -job 3-stage flexible flowshop scheduling problem, the problem is decomposed into n digraphs by removing the dummy node A_d in the superimposed digraph. Each digraph (job) is assigned to an agent who is responsible for scheduling the digraph (job). Therefore, after the problem decomposition and assignment of jobs to the agents, the set of available jobs is set as $SAJ = \{J_1, J_2, \dots, J_n\}$.

Having described agent identification, the game theoretic solution concepts, the driving equations in decision making among agents, and the value functions used in this work are next introduced and explained.

4.2. Modelling of agent cooperation

Before describing the essence of agent cooperation, some discussion and background information in game theory are presented.

4.2.1. Applications of game theory for agent cooperation

Games are characterized by a number of players or decision makers who interact, possibly threaten each other and form groups, take actions under certain conditions, and finally receive some benefits or reward or possibly punishment or loss. The theory can be applied to coordinate systems that involve multiple decision makers with possibly different objectives. The standard procedure in applying game theory is to formulate a model that captures a situation and to investigate the set of outcomes that are consistent with some solution concept. Most of the time, a game is not a description of some physical rules that exist. Most strategic situations lack a clear structure. In this game theoretic analysis based scheduling framework a *planner* or *schedule manager*, is assumed to set the rules of the interactions, and the individuals/agents when confronted with these rules are assumed to take them literally. An assumption underlying this interpretation of interaction is that the scheduling manager can force the agents/players to play the game but cannot enforce the outcome directly. In our model the scheduling manager is responsible for necessary information sharing with the agents, i.e. the winner agent, current schedule/operation assignments, etc. It may also decide whether rescheduling should be performed.

In all game theoretic models the basic entity is a *player* or an *agent*. A player may be interpreted as an individual or a group of individuals making a decision. The words '*agent*' and '*player*' will be used interchangeably throughout this work.

The theory of games is generally divided into two branches, the *cooperative* theory and *non-cooperative* theory. The cooperative model is distinguished from a non-cooperative model primarily by its focus on what groups of players can achieve rather than on what individual players can do and by the fact that it does not consider the details of how groups of players function internally. The outcome of the game is defined by a cooperative joint action of the agents involved in the game. The objective of the scheduling problem in this research is the minimization of maximum completion time of the *overall* schedule. Therefore, we think the cooperative model of the game is appropriate.

The cooperative game theory itself breaks down into two branches, *transferable utility* and *non-transferable utility*, depending on whether or not the players have comparable units of utility and are allowed to make monetary side payments in utility units as an incentive to induce certain strategy choice.

We first define the scheduling problem as an N -person *cooperative transferable utility* (N -TU) game with complete information. The transferability entails that the

players have comparable units of utility. One of the main features of the cooperative games is that the players have freedom to choose the joint strategy. This allows any possible outcome to be in a set of feasible decisions.

4.2.2. Some definitions

For formulation purposes some notations and concepts are introduced next.

Let $n \geq 2$ denote the number of players in a game, numbered from 1 to n , and N denote the set of players: $N = \{1, 2, \dots, n\}$. A coalition, S , is defined as a subset of N , $S \subset N$, and the set of all coalitions is denoted by Ω . By convention we also should define an empty set, $\{\emptyset\}$, as the empty coalition. The set N is called the grand coalition. In the context of scheduling, a player in the game is an agent who has been assigned a job to be scheduled in the system. Correspondingly, a coalition will be a set of agents.

Formulating a game as an *N-TU game* entails specifying the value of the characteristic function $v(S)$ for each coalition $S \in \Omega$, where Ω is the set of all non-empty coalitions from N . The quantity $v(S)$ is a real number assigned to each non-empty coalition $S \subset N$, which may be considered as the value, or power, or worth of coalition S . In addition $v(\emptyset) = 0$ condition must be satisfied.

The Shapley value may be considered as a measure of the power of a player in a *N-TU game*. It is a solution concept in game theory that was proposed by Shapley in 1953 and has found its applications in many voting systems where it is called Shapley–Shubik power index. This principle leads to a unique way to split the payoff from the grand coalition.

The Shapley value $\Phi = \{\phi_1, \phi_2, \dots, \phi_n\}$ for the game players with characteristic function $v(S)$ is calculated by the formula in equation (6):

$$\phi_i(v) = \sum_{\substack{S \subset N \\ i \in S}} \frac{(|S| - 1)!(n - |S|)!}{n!} [v(S) - v(S - \{i\})]. \quad (6)$$

The Shapley value is interpreted as follows. Suppose that all the players are arranged in some order, all orders being equally likely. Then $\phi_i(v)$ is the expected marginal contribution over all orders of player i to the set of players who precede. Note that the sum of the marginal contributions of all players in any ordering is $v(N)$.

Other interesting solution concepts developed in *n-person cooperative game theory* are the *core* and *nucleolus*. Core is not an appropriate solution concept for our problem because the core does not always exist. The nucleolus solution coincides with the Shapley value solution for our problem and related discussion will be provided in the next section.

The description of the scheduling process and the details of decision-making and cooperation are presented next.

4.3. Scheduling and agent cooperation

In a distributed and cooperative scheduling environment, there are feasible scheduling agent candidates and infeasible scheduling agent candidates. The feasible scheduling agent candidates are the ones who have satisfied all their precedence constraints and can schedule their jobs. The infeasible scheduling agent candidates are the ones who have not satisfied all precedence constraints and hence cannot schedule their jobs in the system. Therefore, the game is changing at each stage of scheduling, affecting changes in the set of players and the characteristic function

of the game. The agent entering the game is the one who has the largest power index. In other words it is the one that causes the highest percentage change to the value of the game.

The sets SAJ , NSJ and SJ are defined and initiated once the agents are identified and jobs are assigned to them. The sets are updated at each stage of the scheduling process. The overall agent cooperation during the scheduling is modelled as a game within the game. They are referred as the *outer* and *inner* games. The *outer* game consists of identifying the winning player from the set of agents eligible for scheduling their jobs in the system, or otherwise, identifying the agent eligible for entering the game. An agent with the highest power index, which indicates the highest percentage change in the current schedule, is considered as a winner. The *inner* game is among the players having already entered the game.

Before describing the competition among the agents, we first need to define the value function, i.e. the characteristic function of the game. To define the characteristic function for our scheduling game is to define $v(S)$ as a value associated with the makespan of a schedule, C_{\max} , when S acts as a separate scheduling unit. Formally, it can be defined as in equation (7).

$$v(S) = C_{\max}(x_1, x_2, \dots, x_s), \text{ where } s = |S|. \quad (7)$$

In equation (7), $C_{\max}(x_1, x_2, \dots, x_s)$ is the best attainable makespan of scheduling units involved in the game, when agent a_j implements strategy x_j . The x_j strategy for agent a_j corresponds to the assignment of operations to corresponding machines. The assignment of operations can be modelled as an operations assignment problem. The mixed integer programming (MIP) formulation of an operations assignment problem is provided next.

MIP formulation of operations assignment problem

The MIP formulation of the operations assignment problem solved by individual agents is presented in equations (8) through (12). Before presenting the formulation some notations and variables are introduced.

- K number of stages in the system;
- m_i number of machines at stage $i, i = 1, 2, \dots, k$;
- t_i processing time of operation $O_i, i = 1, 2, \dots, k$;
- T_{ij} completion time on machine $j, j = 1, 2, \dots, m_i$ at stage $i, i = 1, 2, \dots, k$ before the job is scheduled;
- M an arbitrary large number.

Variables

$$x_{ij} = \begin{cases} 1, & \text{if operation } O_i \text{ is assigned to machine } j, i = 1, 2, \dots, k; j = 1, 2, \dots, m_i \\ 0, & \text{otherwise} \end{cases}$$

CT_i completion time of operation O_i at stage $i, i = 1, 2, \dots, k$;

$$\text{Min } Z = CT_k \quad (8)$$

subject to:

$$\sum_{j=1}^{m_i} x_{ij} = 1 \quad \text{for } i = 1, 2, \dots, k, \quad (9)$$

$$CT_{i+1} - CT_i \geq t_{i+1} \quad \text{for } i = 1, 2, \dots, k - 1, \quad (10)$$

$$T_{ij} + t_i x_{ij} - CT_i \leq M(1 - x_{ij}) \quad \text{for } i = 1, 2, \dots, k; j = 1, 2, \dots, m_i, \quad (11)$$

$$x_{ij} = 0 \text{ or } 1; CT_i \quad i = 1, 2, \dots, k; j = 1, 2, \dots, k \text{ are non-negative.} \quad (12)$$

The objective function (8) minimizes the completion time of last operation in the job assigned to an individual agent. Constraint (9) ensures that each operation O_i is assigned to only one machine at stage i . Constraint (10) ensures that the succeeding operations will not start before the completion of its immediately preceding operation. Constraint (11) ensures schedule gaps are wide enough to avoid interference. Constraint (12) ensures integrity and non-negativity.

Note that, in order to define a decision process as a game, we need at least 2 players involved in the process. After initialization of the game, the set of scheduled jobs is empty. This means that there is no coalition of more than 1 player. To initiate the process, at the first stage of scheduling, the winning agent is defined to be one who possesses the 'most important' job to be scheduled. The importance is defined by the length of the time a job requires to finish.

After the first agent enters the inner game, the agents from the feasible agent candidate set are competing with each other to form a coalition with the agent(s) in the inner game. This is called the *estimation* process. The winning agent is the one that has the highest percentage power index. The details are presented below.

Suppose coalition S of agents is in the inner game, having $v(S)$ as its coalition value, which corresponds to the combination schedule by the agents in S . Each agent in the outer game schedules its job with S , treating S as a union. This is very important to remember, because when calculating the Shapley value, the coalition S is treated as a separate decision making unit and its size is equal to 1, i.e. $|S| = 1$. The function values used in calculations of Shapley values are: $v(S)$, $v(\{j\})$, and $v(S \cup \{j\})$.

The power index for agent j is calculated as in equation (13).

$$\begin{aligned} \phi_j(v) &= \frac{(2-1)!(2-2)!}{2!} + [v(S \cup \{j\}) - v(S)] + \frac{(1-1)!(2-1)!}{2!} [v(\{j\}) - v(\emptyset)] \\ &= \frac{1}{2} [v(S \cup \{j\}) - v(S) + v(\{j\})] \end{aligned} \quad (13)$$

The Shapley value for coalition unit S is represented by equation (14). It equal to the remainder of the payoff of the grand coalition, which is $S \cup \{j\}$ in this case.

$$\phi_S(v) = v(S \cup \{j\}) - \phi_j(v). \quad (14)$$

Nucleolus is a solution concept introduced by Schmeidler (1969). Here, they look for a fixed characteristic function $v(S)$, $S \subset N$, and try to find an imputation $x = \{x_1, x_2, \dots, x_n\}$ that minimizes the worst inequity in payoffs. It turns out that the nucleolus and Shapley value solutions are equivalent for our problem and that makes our solution concept double 'secured'. This is because if the nucleolus tries to minimize the maximum inequity payoff, which is $v(\{j\}) - \phi_j(v)$ for player j and

$v(S) - \phi_s(v)$ for unity S then we have inequity payoffs for both as in equations (15) and (16).

$$\begin{aligned} v(\{j\}) - \phi_j(v) &= v(\{j\}) - \frac{1}{2}[v(S \cup \{j\}) - v(S) + v(\{j\})] \\ &= \frac{1}{2}[v(\{j\}) - v(S \cup \{j\}) + v(S)] \end{aligned} \quad (15)$$

$$\begin{aligned} v(\{S\}) - \phi_s(v) &= v(\{S\}) - \frac{1}{2}[v(S \cup \{j\}) + v(S) - v(\{j\})] \\ &= \frac{1}{2}[v(\{j\}) - v(S \cup \{j\}) + v(S)] \end{aligned} \quad (16)$$

Comparing equations (15) and (16) we see that the inequity payoffs are equal for agent j and unity S . Hence the Shapley value solution is also the nucleolus for this game.

The percentage contribution of agent j to the schedule is calculated by the formula in equation (17). It represents the ratio of an individual payoff, $\phi_j(v)$, over the payoff of the grand coalition, $v(S \cup \{j\})$.

$$ShV_j = \frac{\phi_j(v)}{\phi_s(v) + \phi_j(v)} = \frac{\phi_j(v)}{v(S \cup \{j\})}. \quad (17)$$

The agent obtaining the highest percentage contribution in the estimation stage enters the inner game. Once an agent from the outer game enters the inner game, the scheduling stage of the outer game is incremented by 1, the sets of feasible candidate jobs, infeasible jobs, and scheduled jobs are updated, and the competition among agents to enter the inner game starts over again. The pseudo-code of the algorithm carried out in the outer game is presented next.

The outer game procedure

Step 1. Input: Manufacturing system structure and precedence constraints in schedulable jobs.

Step 2. Decompose and initialize $t = 0, SAJ_t, SJ_t, NSJ_t$.

Step 3. For $\forall i \in SAJ_t$

Schedule individual jobs ($v(i) = C_{\max, i}$) by solving the corresponding operations assignment problem (13)–(17).

Step 4. Calculate $k = index(\max_{i \in SAJ_t} \{v(i)\})$.

Step 5. Agent a_k enters the inner game.

Step 6. Update $t = t + 1, SAJ_t, SJ_t, NSJ_t$.

Step 7. If $(SAJ_t = \emptyset)$ Then

GoTo Step 8.

Else

For $\forall i \in SAJ_t$ calculate: $\phi_i(v) = \frac{1}{2}[v(SJ_t \cup \{i\}) - v(SJ_t) + v(\{i\})]$ and

$$ShV_i = \phi_i(v)/v(SJ_t \cup \{i\})$$

$k = index(\max_{i \in SAJ_t} \{ShV_i\})$.

Agent a_k enters inner game.

Update $t = t + 1, SAJ_t, SJ_t, NSJ_t$.

GoTo Inner Game($(S = SJ_t, v(S) = \text{Schedule}(SJ_t))$).

GoTo Step 7.

Step 8. End of Outer Game. Output the overall schedule.

After an agent enters the inner game, it becomes a player in the inner game. The inner game takes care of the cooperation to achieve the best possible outcome, i.e., the minimum makespan. In the inner game, the cooperation mechanism is regulated by the power index (Shapley value) of the players. The purpose of the cooperation is to improve the overall schedule. Hence, it is initiated with an intention to improve the schedule. Some conditions are used to check for initiating the cooperation:

- (1) the makespan is close enough to the estimated upper bound; and
- (2) the starting of an operation scheduled by an agent is strictly greater than the completion time of preceding operations. (Basically, this fact indicates that the agent is delaying his job because of other agents. Hence, it may consider cooperation with others.)

After determining the need for cooperation, if any, the process is regulated as follows. Each agent in the inner game considers rescheduling its job without changing any assignment made by other agents in the game. The one who has more power, which is defined by its contribution to the game, gets the right for final rescheduling. The power of agents is calculated by the Shapley value of the player, similar to the one in the outer game.

This process is iterative and continues until agents find that there is no need for further cooperation and hence there is no need for further rescheduling. The following conditions are recommended to be checked for the termination of rescheduling:

- (1) the makespan is close enough to the estimated lower bound;
- (2) the difference between two successive makespans is in a predefined *small* range; and
- (3) a predefined number of iterations is reached.

Other appropriate terminating conditions can also be developed depending on the objectives, system, etc.

When the cooperation is over, the inner game reaches the ‘*stabilization*’ stage. After the ‘*stabilization*’ of the inner game, the outer game resumes and new agents enter the inner game. The pseudo-code of the algorithm for implementing the inner game is presented next.

The inner game procedure

Step 1. Input: Coalition $S \subseteq N$, and Payoff $v(S)$

Step 2. For $\forall i \in S$

Calculate: $v(S'_i) = v((S - \{i\}) \cup \{i^*\})$;

$(i^*$ is a new solution of agent a_i by solving the corresponding operations assignment problem (13)–(17))

$\phi_i(v) = \frac{1}{2} [v(S'_i) - v(S) + v(\{i\})]$ and

$ShV_i = \phi_i(v)/v(S'_i)$.

Step 3. Calculate $k = index(\max_{i \in S} \{ShV_i\})$.

Step 4. Agent a_k re-schedules job k .

Step 5. If (Terminate Rescheduling) Then

GoTo Step 6.

Else

GoTo Step 2.

Step 6. Output: Inner Game is stabilized.

N	m_1, m_2, m_3	C_{AB}	C_{LB}	C_{AB}/C_{LB}	SE_{\min}	SE_{\max}
30	2,2,2	987.56	856.24	1.15	1.20	1.73
	2,2,3	915.92	835.44	1.10	1.16	1.54
	2,3,2	960.32	850.20	1.13	1.14	1.66
	2,3,3	849.48	812.76	1.05	1.10	1.52
	3,2,2	970.20	819.52	1.18	1.16	1.55
	3,2,3	862.92	762.74	1.13	1.17	1.54
	3,3,2	931.76	801.50	1.16	1.10	1.53
	3,3,3	704.16	583.71	1.21	1.27	1.72
	50	2,2,2	1543.36	1377.10	1.12	1.22
2,2,3		1441.24	1337.90	1.08	1.16	1.57
2,3,2		1479.56	1365.96	1.08	1.13	1.74
2,3,3		1344.24	1310.44	1.03	1.06	1.57
3,2,2		1501.16	1331.82	1.13	1.15	1.57
3,2,3		1370.76	1263.54	1.08	1.17	1.58
3,3,2		1449.36	1296.88	1.12	1.05	1.55
3,3,3		1071.24	926.65	1.16	1.27	1.77

C_{AB} – makespan obtained by applying agent-based approach;

C_{LB} – lower bound on makespan;

SE_{\min} – the best obtained by Soewandi and Elmaghraby (2001); and

SE_{\max} – the words result obtained by Soewandi and Elmaghraby (2001).

Table 1. The comparison results for the first type problems (balanced load).

An illustrative example is presented in the Appendix to demonstrate the application of the game procedure.

5. Performance evaluation

To evaluate the performance of the agent-based approach, solutions to the n -job 3-stage flexible flowshop scheduling problem generated by the agent-based approach were compared with those by the heuristics of Soewandi and Elmaghraby (2001). We generated three types of problems as described in Soewandi and Elmaghraby (2001). The problems were generated as follows: (i) *balanced load* - the processing times at all stages follow an uniform distribution from interval $[1, 100]$, i.e., $P_{j,1}, P_{j,2}, P_{j,3} \sim U(1, 100)$; (ii) *light load on stage 1* – the processing times at stage 1 follow a uniform distribution from interval $[1, 20]$ and the processing times at the second and third stages follow a uniform distribution from interval $[1, 100]$, i.e., $P_{j,1} \sim U(1, 20), P_{j,2}, P_{j,3} \sim U(1, 100)$; (iii) *light load on stage 2* – the processing times at the stage 2 follow a uniform distribution from the interval $[1, 20]$ and the processing times at first and third stages follow a uniform distribution from interval $[1; 100]$, i.e. $P_{j,1}, P_{j,3} \sim U(1, 100), P_{j,2} \sim U(1, 20)$. The number of jobs was 30 and 50. For each type of the problem, 25 instances were generated. The number of machines in every stage (m_1, m_2, m_3) was a combination of $m_k = 2, 3$ for $k = 1, 2, 3$. In total, the number of tested problems is $3 \times 8 \times 2 \times 25 = 1200$.

For each problem, the ratio (C_{AB}/C_{LB}) for each instance was computed and averaged over the 25 instances. The data in tables 1 through 3 represent the obtained experimental results. As mentioned in section 4.3, the MIP formulation of the operations assignment problem is used by individual agents to schedule their jobs in the system. The computational complexity (cc) of the presented agent-based scheduling method depends on the computational complexity of an algorithm (cc_{sub}) applied to solve each individual agents' scheduling problems. For example, in an environment of n agents, where r reschedulings are required, the computational complexity would

N	m_1, m_2, m_3	C_{AB}	C_{LB}	C_{AB}/C_{LB}	SE_{\min}	SE_{\max}
30	2,2,2	886.60	823.68	1.08	1.02	1.16
	2,2,3	796.64	764.23	1.04	1.03	1.17
	2,3,2	865.96	782.66	1.11	1.01	1.15
	2,3,3	618.24	554.33	1.12	1.04	1.25
	3,2,2	888.52	823.68	1.08	1.01	1.10
	3,2,3	797.36	764.22	1.04	1.02	1.13
	3,3,2	865.52	782.66	1.11	1.01	1.11
	3,3,3	614.80	553.87	1.11	1.02	1.18
	50	2,2,2	1394.76	1336.92	1.04	1.02
2,2,3		1299.16	1275.98	1.02	1.02	1.20
2,3,2		1356.12	1269.96	1.07	1.01	1.17
2,3,3		956.28	894.947	1.07	1.04	1.28
3,2,2		1397.40	1336.92	1.05	1.01	1.12
3,2,3		1299.44	1275.98	1.02	1.02	1.13
3,3,2		1356.24	1269.96	1.07	1.01	1.10
3,3,3		951.56	894.613	1.06	1.02	1.19

Table 2. The comparison results for the second type problems (light load on stage 1).

N	m_1, m_2, m_3	C_{AB}	C_{LB}	C_{AB}/C_{LB}	SE_{\min}	SE_{\max}
30	2,2,2	890.96	829.82	1.07	1.13	1.68
	2,2,3	813.44	787.39	1.03	1.09	1.48
	2,3,2	891.60	829.82	1.07	1.12	1.68
	2,3,3	812.64	787.37	1.03	1.08	1.49
	3,2,2	848.40	769.48	1.10	1.10	1.52
	3,2,3	619.56	559.77	1.11	1.17	1.72
	3,3,2	848.60	769.48	1.10	1.09	1.52
	3,3,3	613.36	559.33	1.10	1.20	1.73
	50	2,2,2	1408.52	1349.82	1.04	1.10
2,2,3		1312.84	1292.36	1.02	1.05	1.49
2,3,2		1409.56	1349.82	1.04	1.13	1.70
2,3,3		1311.28	1292.36	1.01	1.05	1.49
3,2,2		1358.76	1269.60	1.07	1.07	1.54
3,2,3		966.56	905.40	1.07	1.15	1.74
3,3,2		1358.08	1269.60	1.07	1.06	1.56
3,3,3		1065.64	905.04	1.18	1.16	1.75

Table 3. The comparison results for the third type problems (light load on stage 2).

be $cc = n \times r \times cc_{sub}$. All of our computations were performed on a Pentium IV, 2 GHz PC computer and the computational time was in the range of 0.5–3.0 seconds.

For the first type of the problems, when the stages have balanced load (table 1) the agent-based scheduling approach outperforms SE heuristics ($(C_{AB}/C_{LB}) < SE_{\min}$) for almost all system structures. Also, the ratio values are far better than SE_{\max} .

The results for the second type of the problems where the stage 1 task is dominated by the other two stages are presented in table 2. The heuristics SE give very good solutions for this type of problem. The average error is less than 4%. For some of the cases, the agent-based approach obtained comparative error bounds. However, in general, the SE heuristic outperformed the agent-based approach for this type of the problem. The reason that the SE heuristic gives better solution than the agent-based approach for this type of problem can be explained as follows. First, this type of

problem represents a very special case of the n -job 3-stage flexible flowshop scheduling problem, which can be approximated as a 2-stage flowshop scheduling, since the first stage on average has five times less load than the other two stages. Second, the heuristic by Soewandi and Elmaghraby (2001) is developed based on Johnson's algorithm, which provides optimal solution for 2-machine flowshop problems, and is able to obtain solutions very close to optimum. Third, the authors reversed the role of stage 1 and stage 3, in the problem, and applied the heuristic to the new problem. In this case, Johnson's algorithm is applied to stages 2 and 3 of the original problem, and stage 1 has a lower load than stages 2 and 3 and hence will not have much effect on the final schedule. Even for this special case of the n -job 3-stage flexible flowshop scheduling problem, the solutions obtained by the agent-based approach are competitive. It is argued that the agent-based approach represents a general and powerful tool for solving the manufacturing system scheduling problems.

The results obtained for the third type of the problems when the stage 2 task is dominated by the other two stages, is presented in table 3. As for the type 1 problems, the agent-based approach outperformed the best SE heuristics ($(C_{AB}/C_{LB}) < SE_{\min}$) for the majority of the system structures and for all the cases, the ratio (C_{AB}/C_{LB}) is strictly better than SE_{\max} .

6. Conclusions

In this paper, an agent-based manufacturing system scheduling methodology is presented for solving the n -job 3-stage flexible flowshop scheduling problem. The agent-based scheduling methodology is general and suitable for solving large-scale manufacturing systems scheduling problems. The key in solving the scheduling problem with the agent-based approach is the cooperation and coordination of the agents. For coordination of the agents, classical tools for multiple decision makers developed in games theory are applied for problem solving. In particular, the cooperative model of games is considered, where agents form coalitions to maximize the payoff from their cooperation. The power index of an agent is defined by the Shapley value, which associates the schedule completion time in the system achieved by the agent. The overall scheduling process is an integration of defined inner and outer games where the outer game is among the agents eligible to schedule their jobs in the system and the inner game is among the agents eligible to *re-schedule* their jobs in the system.

To evaluate the performance of the agent-based scheduling approach for solving the n -job 3-stage flexible flowshop scheduling problem, computational experiments were conducted. The developed agent-based scheduling framework was coded in C++ and the makespans of the schedules generated by the agent-based approach were calculated for a set of testing problems. The error bounds for agent-based solutions were calculated as the ratios of the makespan obtained by the agent-based approach over the lower bound computed using the formulas developed by Soewandi and Elmaghraby (2001).

Three types of problems were tested: (i) balanced load at each processing stage; (ii) light load at stage 1; (iii) light load at stage 2. The computational results showed that the agent-based approach outperforms the heuristics developed by Soewandi and Elmaghraby (2001) for most of the tested problems.

The work presented in this paper can be viewed as an important step in developing powerful tools for solving distributed scheduling problems for agile manufacturing. There are some research issues that are worthy of future research. First, it

would be interesting to study the effect of different negotiation models on the scheduling solution. The winner in the negotiation process presented in this paper is the one that has the largest percentage contribution in the best possible schedule obtained at the current stage. Other measurement values can be considered, such as the largest/smallest marginal contribution to the model, shortest makespan obtained, etc. Second, it would be interesting to identify problem instances, based for example on the processing load of the machines at different stages on which the developed negotiation models work better. The other research possibility would be the investigation of the agent-based scheduling approach for solving general k -stage flexible flowshop scheduling problems.

Appendix. Illustrative example

Consider the set of three jobs $\{j_1, j_2, j_3\}$, with precedence structure as presented in figure 2, to be scheduled in the three-stage flexible manufacturing system of structure $\{m_1=2, m_2=1, m_3=1\}$.

Following the supercomposition procedure, a dummy job j_4 is added to the set of these jobs, with a completion time equal to zero, and jobs j_1, j_2 , and j_3 proceed to j_4 . An agent is assigned to each job and the set of agents is created as $\{a_1, a_2, a_3, a_4\}$.

The course of the scheduling is presented below. For simplicity, we omit the *pure* scheduling part and assume that the values $v(1), v(2), v(3), v(1,2), v(3,2), v(1,2,3), v(1^*, 2), v(1, 2^*), v(1^*, 2, 3), v(1, 2^*, 3)$ and $v(1, 2, 3^*)$ represent the scheduled completion time obtained by individual agents after solving the operations assignment problems in equations (13) through (17).

The abbreviations OG and IG stand for Outer Game and Inner Game respectively.

OG_Step 1. Input the manufacturing system structure and the precedence relations of jobs.

OG_Step 2. Initialize the system as follows:

$$t = 0, SAJ_0 = \{j_1, j_2, j_3\}, SJ_0 = \{\emptyset\}, NSJ_0 = \{j_4\}.$$

OG_Step 3. Each agent in SAJ_t schedules its job and the values $v(1)=15, v(2)=18$, and $v(3)=11$ are obtained.

OG_Step 4. $k = \text{index}(\max_{i \in SAJ_0} \{v(i)\}) = \text{index}(\max\{15, 18, 11\}) = 2$.

OG_Step 5. Agent a_2 enters the inner game.

OG_Step 6. Update $t = 1, SAJ_1 = \{j_1, j_3\}, SJ_1 = \{j_2\}, NSJ_1 = \{j_4\}$.

OG_Step 7. $SAJ_t \neq \emptyset$ therefore

each agent that has a job in SAJ_1 schedules its job and the values $v(1,2)=25$ and $v(2,3)=21$ are obtained.

$$\begin{aligned} \phi_1(v) &= \frac{1}{2} [v(SJ_1 \cup \{1\}) - v(SJ_1) + v(1)] = \frac{1}{2} [v(1, 2) - v(2) + v(1)] \\ &= \frac{1}{2} [25 - 18 + 15] = 11 \end{aligned}$$

$$ShV_1 = \phi_1(v)/v(SJ_1 \cup \{1\}) = 11/25$$

$$\begin{aligned} \phi_3(v) &= \frac{1}{2} [v(SJ_1 \cup \{3\}) - v(SJ_1) + v(3)] = \frac{1}{2} [v(2, 3) - v(2) + v(3)] \\ &= \frac{1}{2} [21 - 18 + 11] = 7 \end{aligned}$$

$$ShV_3 = \phi_3(v)/v(SJ_1 \cup \{3\}) = 7/21$$

$$k = \text{index}(\max_{i \in SAJ_1} \{ShV_i\}) = \text{index}(\max\{11/25; 7/21\}) = 1.$$

Agent a_1 enters the inner game.

Update $t = 2$, $SAJ_2 = \{j_3\}$, $SJ_2 = \{j_1, j_2\}$, $NSJ_2 = \{j_4\}$.

GoTo the Inner Game($S = SJ_2$, $v(S) = \text{Schedule}(SJ_2)$).

IG_Step 1. Input: $S = SJ_2 = \{1, 2\}$, $v(S) = \text{Schedule}(SJ_2) = 25$.

IG_Step 2. Each agent in SJ_2 re-schedules its job and the values

$$v(S'_1) = v((S - \{1\}) \cup \{1^*\}) = v(1^*, 2) = 25 \text{ and}$$

$$v(S'_2) = v((S - \{2\}) \cup \{2^*\}) = v(1, 2^*) = 24 \text{ are obtained}$$

(i^* is a new solution of agent a_i by solving the corresponding operations assignment problem (13)–(17))

$$\phi_1(v) = \frac{1}{2} [v(S'_1) - v(S) + v(1)] = \frac{1}{2} [25 - 25 + 15] = 7.5 \text{ and}$$

$$ShV_1 = \phi_1(v)/v(S'_1) = 7.5/25$$

$$\phi_2(v) = \frac{1}{2} [v(S'_2) - v(S) + v(2)] = \frac{1}{2} [24 - 25 + 18] = 8.5 \text{ and}$$

$$ShV_2 = \phi_2(v)/v(S'_2) = 8.5/24.$$

IG_Step 3.

$$k = \text{index}(\max_{i \in SJ_2} \{ShV_i\}) = \text{index}(\max\{7.5/25; 8.5/24\}) = 2.$$

IG_Step 4. Agent a_2 re-schedules job 2.

IG_Step 5. Terminate rescheduling GoTo IG_Step 6.

IG_Step 6. Inner Game is stabilized.

OG_Step 8. $SAJ_t \neq \emptyset$ therefore

each agent that has a job in SAJ_t schedules its job and the value $v(1,2,3) = 31$ is obtained.

$$\begin{aligned} \phi_3(v) &= \frac{1}{2} [v(SJ_2 \cup \{3\}) - v(SJ_2) + v(3)] = \frac{1}{2} [v(1, 2, 3) - v(1, 2) + v(3)] \\ &= \frac{1}{2} [31 - 24 + 11] = 9 \end{aligned}$$

$$ShV_3 = \phi_3(v)/v(SJ_2 \cup \{3\}) = \phi_3(v)/v(1, 2, 3) = 9/31$$

$$k = \text{index}(\max_{i \in SAJ_2} \{ShV_i\}) = \text{index}(\max\{9/31\}) = 3.$$

Agent a_3 enters the inner game.

Update $t = 3$, $SAJ_3 = \{j_4\}$, $SJ_3 = \{j_1, j_2, j_3\}$, $NSJ_3 = \{\emptyset\}$.

GoTo the Inner Game ($S = SJ_3$, $v(S) = \text{Schedule}(SJ_3)$).

IG_Step 1. Input: $S = SJ_3 = \{1, 2, 3\}$, $v(S) = \text{Schedule}(SJ_3) = 31$.

IG_Step 2. Each agent in SJ_3 re-schedules its job and the values

$$v(S'_1) = v((S - \{1\}) \cup \{1^*\}) = v(1^*, 2, 3) = 31,$$

$$v(S'_2) = v((S - \{2\}) \cup \{2^*\}) = v(1, 2^*, 3) = 31, \text{ and}$$

$$v(S'_3) = v((S - \{3\}) \cup \{3^*\}) = v(1, 2, 3^*) = 31 \text{ are obtained}$$

this means no agent can offer an improvement in the current schedule therefore Inner Game is *naturally* stabilized. (GoTo IG_Step 6).

IG_Step 6. Inner Game is stabilized. Since dummy job j_4 will not change the schedule, the output of the inner game at stage 3 will be the final schedule after $SAJ_t = \emptyset$.

OG_Step 8. End of the Outer Game. Output the overall schedule with completion time 31.

References

- AUMANN, R., 1959, Acceptable points in general cooperative n -person games. In: *Contributions to the Theory of Games*, Vol. 4, (Princeton: Princeton University Press).
- BAKER, A. D., 1991, Manufacturing control with a market-driven contract net. PhD thesis, Rensselaer Polytechnic Institute, NY.
- BUSSMANN, S. and SCHILD, K., 2001, An agent-based approach to the control of flexible production systems. *IEEE Symposium on Emerging Technologies and Factory Automation, ETFA*, **2**, 481–488.
- BUTLER, J. and OHTSUBO, H., 1992, ADDYMS: Architecture for distributed dynamic manufacturing scheduling. In A. Famili, D. S. Nau and S. H. Kim (eds), *Artificial Intelligence Applications in Manufacturing* (The AAAI Press), 199–214.
- BYUNG-IN, K., GRAVES, R. J., HERAGU, S. S. and St ONGE, A., 2002, Intelligent agent modeling of an industrial warehousing problem. *IIE Transactions*, **34**(2), 601–612.
- CAMPBELL, H. G., DUDEK, R. A. and SMITH, M. L., 1970, A heuristic algorithm for the n -job m -machine sequencing problem. *Management Science*, **16**, B630–B637.
- CARLIER, J., 1987, Scheduling jobs with release dates and tails on identical machines to minimize makespan. *European Journal of Operational Research*, **29**, 298–306.
- CHEN, B., GLASS, C. A., POTTS, C. N. and STRUSEVICH, V. A., 1996, A new heuristic for three-machine flowshop scheduling. *Operations Research*, **44**(6), 891–898.
- DECKER, K. S., 1987, Distributed problem solving techniques: a survey. *IEEE Transactions on Systems, Man and Cybernetics*, **17**(5), 729–740.
- DUFFIE, N. A. and PIPER, R. S., 1986, Non-hierarchical control of manufacturing systems. *Journal of Manufacturing Systems*, **5**(2), 137–139.
- FISCHER, K., MULLER, J. P., PISCHEL, M. and SCHIER, D., 1995, A model for cooperative transportation scheduling. *Proceedings of ICMAS'95*, San Francisco, CA (AAAI Press/MIT Press), pp. 109–116.
- FORDYCE, K. and SULLIVAN, G. G., 1994, Logistics management system (LMS): integrating decision technologies for dispatch scheduling in semiconductor manufacturing. In M. Zweben and M. S. Fox (eds), *Intelligent Scheduling* (San Francisco, CA: Morgan Kaufmann), pp. 473–516.
- GALLIERS, J. R., 1988, A theoretical framework for computer models of cooperative dialogue. Acknowledging multi-agent conflict. PhD thesis, Open University, UK.
- GAREY, M. R., JOHNSON, D. S. and SAHNI, S., 1976, Fowshop and jobshop schedules: complexity and approximation. *Mathematics of Operations Research*, **1**, 117–127.
- GENESERETH, M. R. and KETCHPEL, S. P., 1994, Software agents. *Communications of the ACM*, **37**(7), 48–53.
- GUINET, A. and SOLOMON, M., 1996, Scheduling hybrid flowshop to minimize maximum tardiness or maximum completion time. *International Journal of Production Research*, **34**(6), 1643–1654.
- JACKSON, J. R., 1955, Scheduling a production line to minimize maximum tardiness. *Research No. 43, Management Science Research*, University of California, Los Angeles.
- JENNINGS, N. R., VARGA, L. Z., AARNTS, R. P., FUCHS, J. and SKAREK, P., 1993, Transforming standalone expert systems into a community of cooperating agents. *International Journal of Engineering Applications of Artificial Intelligence*, **6**(4), 317–331.
- JENNINGS, N. R., CORERA, J., LARESGOITI, I., MAMDANI, E. H., PERRIOLAT, F., SKAREK, P. and VARGA, L. Z., 1995, Using ARCHON to develop real word DAI applications for electricity transportation management and particle accelerator control. *IEEE Expert Special Issue on Real World Applications of DAI*.
- JOHNSON, S. M., 1954, Optimal two and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, **1**, 61–68.
- KLUSCH, M. and GERBER, A., 2002, Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, **17**(3), 42–47.
- KNOTTS, G., DROR, M. and BRUCEN, C. H., 2000, Agent-based project scheduling. *IIE Transactions*, **32**(5), 387–401.

- KUTANOGLU, E. and WU, S. D., 2002, An incentive compatible mechanism for distributed resource planning. Available at <http://www.lehigh.edu/~sdw1/papers2.htm>.
- LEE, C.-Y. and VAIRAKTARAKIS, G., 1994, Minimizing makespan in hybrid flowshops. *Operations Research Letters*, **16**, 149–158.
- LIN, G. Y.-J. and SOLBERG, J. J., 1992, Integrated shop floor control using autonomous agents. *IIE Transactions: Design and Manufacturing*, **24**(3), 57–71.
- LIU, J.-S. and SYCARA, K. P., 1998, Multiagent coordination in tightly coupled task scheduling. In M. N. HUHNS and M. P. Singh (Eds), *Readings in Agents* (San Francisco, CA: Morgan Kaufmann), pp. 164–171.
- MINSKY, M., 1986, *The Society of Mind* (New York: Simon and Schuster).
- MORLEY, R. E. and SCHELBERG, C., 1993, An analysis of a plant specific dynamic scheduler. *Proceedings of the NSF Workshop on Dynamic Scheduling*, Cocoa Beach, Florida.
- NAWAZ, M., ENSCORE, Jr. E. E. and HAM, I., 1983, A heuristic algorithm for the m -machine n -job flow-shop sequencing problem. *OMEGA International Journal of Management Science*, **11**, 91–95.
- OW, P. S. and SMITH, S. F., 1988, A cooperative scheduling system. *Proceedings of the 2nd International Conference on Expert Systems and the Leading Edge in Production Planning and Control*, pp. 43–56.
- PARUNAK, H. V. D., 1995, Applications of distributed artificial intelligence in industry. In G. M. P. O'Hare and N. R. Jennings (Eds), *Foundations of Distributed Artificial Intelligence* (Wiley).
- PARUNAK, V. D., BAKER, A. D. and CLARK, S. J., 1997, AARIA agent architecture: an example of requirements-driven agent-based system design. *Proceedings of the International Conference on Autonomous Agents*, pp. 482–483.
- PARUNAK, V. D., BAKER, A. D. and CLARK, S. J., 2001, AARIA agent architecture: from manufacturing requirements to agent-based system design. *Integrated Computer-Aided Engineering*, **8**(1), 45–58.
- RABELO, R. J. and CAMARINHA-MATOS, L. M., 1994, Negotiation in multi-agent based dynamic scheduling. *Journal on Robotics and Computer Integrated Manufacturing*, **11**(4), 303–310.
- RABELO, R. J., CAMARINHA-MATOS, L. M. and AFSARMANESH, H., 1999, Multi-agent-based agile scheduling. *Robotics and Autonomous Systems*, **27**, 15–28.
- RAMAMRITHAM, K. and STANKOVIC, J. A., 1984, Dynamic task scheduling in hard real-time distributed systems. *IEEE Software*, 65–75.
- RAPOPORT, A., 2001, *N-Person Game Theory* (Ann Arbor, MI: University of Michigan).
- ROSENSCHEIN, J. S. and GENESERETH, M. R., 1985, Deals among rational agents. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, CA, pp. 91–99.
- SANDHOLM, T. and LESSER, R. V., 1997, Coalition among rationally bounded agents. *Artificial Intelligence*, **94**, 99–137.
- SANDHOLM, T., LARSON, K., ANDERSSON, M., SHEHORY, O. and TOHMÉ, F., 1999, Coalition structure generation with the worst case guarantees. *Artificial Intelligence*, **111**, 209–238.
- SCHMEIDLER, D., 1969, The nucleolus of a characteristic function game. *SIAM Journal of Applied Mathematics*, **17**, 1163–1170.
- SCHWUTTKE, U. M. and QUAN, A. G., 1993, Enhancing performance of cooperating agents in real time diagnosis systems. *Proceedings of 13th International Joint Conference on Artificial Intelligence*, Chamberry, France, pp. 332–337.
- SHAPLEY, L., 1953, A value of n person games. In H. Kuhn and A. Tucker (Eds), *Contributions to the Theory of Games II, Annals of Mathematics Studies* 28, (Princeton: Princeton University Press), pp. 307–317.
- SHEHORY, O. and KRAUS, S., 1999, Feasible formation of coalitions among autonomous agents in nonsuperadditive environments. *Computational Intelligence*, **15**(3), 218–251.
- SHEN, W. and NORRIE, D. H., 1998, An agent-based approach for dynamic manufacturing scheduling. *Working Notes of the Agent-Based Manufacturing Workshop*, Minneapolis, MN.
- SHEN, W. and NORRIE, D. H., 1999, Agent-based systems for intelligent manufacturing: a state of the art survey. *Knowledge and Information Systems, an International Journal*, **1**(2), 129–156.

- SICHMAN, J. and DEMAZEAU, Y., 1992, When can knowledge-based systems be called agents? *Proceedings IX Brazilian Symposium on Artificial Intelligence*. Rio de Janeiro, Brazil, 5–8 October.
- SMITH, R. G., 1980, The contract-net protocol: high level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, **29**(12), 1104–1113.
- SMITH, R. G. and DAVIS, R., 1981, Framework for cooperation in distributed problem solving. *IEEE Transactions on System, Man and Cybernetics*, **11**(1), 61–70.
- SMITH, R. G. and DAVIS, R., 1983, Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, **20**, 63–109.
- SOEWANDI, H. and ELMAGHRABY, S. E., 2001, Sequencing three-stage flexible flowshops with identical machines to minimize makespan. *IIE Transactions*, **33**, 985–993.
- SOUSA, P. and RAMOS, A., 1997, A dynamic scheduling Holos for manufacturing systems. *Proceedings of the Second World Congress on Intelligent Manufacturing Processes and Systems*. Budapest, Hungary, 10–13 June.
- SYCARA, K., ROTH, N., SADEH, N. and FOX, M., 1991a, Distributed constrained heuristic search. *IEEE Transactions on Systems, Man and Cybernetics*, **21**(6), 1446–1461.
- SYCARA, K., ROTH, N., SADEH, N. and FOX, M., 1991b, Resource allocation in distributed factory scheduling. *IEEE Expert*, **6**(1), 29–40.
- TOWNSEND, D. W., 1977, Sequencing n job on m machines to minimize tardiness: a branch and bound solution. *Management Science*, **23**, 1016–1019.
- VARGA, L. Z., JENNINGS, N. R. and COCKBURN, D., 1994, Integrating intelligent systems into a cooperating community for electricity distribution management. *International Journal of Expert Systems with Applications*, **7**(4), 563–579.
- WEIHMYER, R. and VELTHUISEN, H., 1994, Application of distributed AI and cooperative problem solving to telecommunications. In J. Liebowitz and D. Prereau (eds), *AI Approaches to Telecommunications and Network Management* (IOS Press).
- WOOLDRIDGE, M. and JENNINGS, N. R., 1995, Intelligent agents: theory and practice. *Knowledge Engineering Review*, **10**(2), 115–152.
- WU, S. H., FUH, J. Y. H. and NEE, A. Y. C., 2002, Concurrent process planning and scheduling for distributed virtual manufacturing. *IIE Transactions*, **34**(1), 77–89.