A multi Agent System Based on Modified Shifting Bottleneck and Search Techniques for Job Shop Scheduling Problems

M.H Karimi Gavareshki [i] and M.H Fazel Zarandi [ii]*

Received 29 July 2007; received in revised 12 April 2009; accepted 6 May 2009

ABSTRACT

This paper presents a multi agent system for the job shop scheduling problems. The proposed system consists of initial scheduling agent, search agents, and schedule management agent. In initial scheduling agent, a modified Shifting Bottleneck is proposed. That is, an effective heuristic approach and can generate a good solution in a low computational effort. In search agents, a hybrid search approach is presented. The schedule management agent can manage the system. Finally, the proposed agent based system is tested and validated by some benchmark problems. The results show the superiority of the proposed system in terms of makespan minimization and CPU times.

KEYWORDS

Job shop scheduling, multi agent system, shifting Bottleneck , search technique.

1. INTRODUCTION

The job shop scheduling problem with which we are concerned can be described as follows: The problem consists of scheduling the jobs on the machines with the objective to minimize the makespan. Any schedule is subjected to two constraints: (i) the precedence of the operations on each job must be respected; (ii) once a machine starts processing an operation it cannot be interrupted, and each machine can process at most one operation at a time.

The job shop scheduling problems are among the hardest combinatorial optimization problems, and are strongly NP-complete [16]. These problems have been widely studied in the literature by various methods: heuristics, shifting Bottleneck , constraint propagation techniques, tabu search, simulated annealing, genetic algorithms, neural networks, etc.

Most of these methods encounter great difficulties when they are applied to real situations, because these scheduling methods use simplified theoretical models and are essentially centralized in the sense that all the computations are carried out in a central computing unit/center. These approaches are therefore inflexible, expensive and slow to satisfy real world scheduling problems. In particular, these approaches are not robust enough to accommodate the dynamic nature of the manufacturing scheduling problem.

Within the past decade, a number of researchers have applied agent technology in attempts to resolve the scheduling problems [27]. The studies undertaken in distributed scheduling may be classified into three categories: hierarchical, heterarchical, and centralized control [29]. In a hierarchical approach like distributed asynchronous system DAS, the system is co-ordinated level by level and the corresponding parts are synchronized [8]. In the heterarchical approach, the systems are co-ordinated by allowing the agents to negotiate amongst themselves.[2] Lastly, the agents can be co-ordinated by a single module utilizing centralized control. Talukdar and Murthy proposed the teams of autonomous agents (ATeams) to solve large combinatorial optimization problems using a multi agent based distributed problem solving method where the agents asynchronously build shared solutions [28], [23]. This method allows the system either to be centrally controlled or decentralized. In an ATeams, a collection of agents co-operates by sharing solutions through a common memory. The architecture is asynchronous and the agents are autonomous. Reference[28] reports success on a number of different problems. Aydin tested the ATeams on the job shop scheduling problem. He shows that the results of ATeams are better than individual agent but that it depends on the portfolio of agents and on

i M.H Karimi Gavareshki , Ph.D., Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran , P.O. Box: 15875-4413
ii * Corresponding Author, M.H Fazel Zarandi, Associate Professor, Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran, P.O. Box: 15875-4413. Email: mh_karimi@aut.ac.ir, zarandi@aut.ac.ir

the way in which the memory is managed [4].

This paper present a new multi agent system based on Ateams by different framework for the job shop scheduling problems. The proposed system consists of initial scheduling agent, search agents and schedule management agent. Moreover, this paper presents a modified Shifting Bottleneck in initial scheduling agent. That is an effective heuristic approach and generates a good solution in a low computational effort. Also, the paper presents a hybrid search approach in search agents. One of the contributions of this paper is presenting a modified Shifting Bottleneck approach for job shop scheduling that is used in the proposed multi agent system. Subproblem solution procedure and reoptimization are two important factors in SB approach that can increase computational efforts. In large scale problems, we need effective procedures to decrease the computational efforts. This paper presents a modified Schrage algorithm for single machine scheduling problems with heads and tails that is an effective subproblem solution procedure. Then we present a heuristic approach for job shop scheduling that resolve reoptimization difficulties in SB. Finally, the proposed multi agent system is tested and validated by some benchmark problems. Experimental results show the superiority of our system.

The rest of this paper is organized as follows: Section 2 presents an overview of multi agent systems. In section 3 the proposed multi agent system is presented. In Section 4, experimental results of the proposed system in comparison with another system are presented. Finally, conclusions and future works are presented in Section 5.

2. MULTI AGENT SYSTEMS

In general, an agent is a computer system that is situated in some environment, and that is capable of flexible and autonomous action in this environment in order to meet its design objectives. By flexible we mean that the system must be responsive, proactive, and social [33]. Various definitions have been proposed for the term multi-agent system (MAS). MAS are a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver. Ferber defines MAS as a system composed of a population of autonomous agents, which interact with each other to reach common objectives, while simultaneously each agent pursues individual objectives [14]. Various interesting features of agents have been proposed and defined in the literature. However, some of them are more important and useful in agent-based manufacturing, e.g., autonomous, cooperative, proactive, and adaptive [27].

As shown in Figure1, the basic structure of an agent consists of the following three modules, namely, the manager module, message processor, and communication protocol [36] :

- Manager module: This is responsible for interacting with other agents, and executing the assigned tasks with a manager unit, a knowledge base, a method based consisting of the required operations, a database, and an agenda, which is a list of the scheduled tasks.
- Message processor: This module stores and processes the messages received and the tasks assigned from the users or other agents.
- Communication protocol: This guarantees the continuity and compatibility of communication between the agents using a unified communication language and protocol.

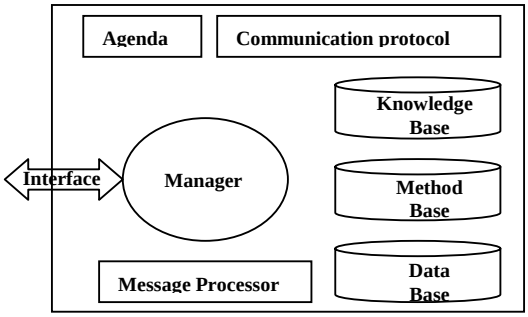

Figure1. Basic structure of an agent.

An A-Team is a team of software agents that cooperate to solve a problem by dynamically evolving a shared population of solutions [23], [28]. Reference [23] proposes architecture of autonomous agent operating asynchronously on a shared population of solution attempts, which they call “ATeams”. In the basic architecture, each agent is completely independent from the rest, and operates by selecting a solution from the memory, carrying out some operations on that solution, and then placing it back in the memory. Co-operation is thus archived by sharing solutions. The population of solution is controlled by a subset of destroyer agents, which evaluate solutions according to certain criteria and remove unwanted solutions. The organization of the agents is that loose-agents may appear and disappear from the team without penalty, may be widely distributed and do not communicate directly with other agents. An instance of ATeam architecture is shown in Figure2. Here, the system consists of a team of agents and a single memory, which has particular communications with each agent [4].

ATeams are, in many respects, similar to blackboard system, in that a collection of processes co-operates to solve problems by posting the results of actions to a shared memory (or blackboard). However, there are some differences. In the blackboard systems, the problem solving process is typically centrally controlled, with a control process deciding which of the available

knowledge source should be activated at which point. Blackboards are typically structured to suit a particular problem, being hierarchically sub-divided, with problems also being sub-divided and sub-problems combined in pre-determined ways. In a basic ATeam, there is no control and each agent operates without knowledge of the others. The memory is typically on a single level [4].

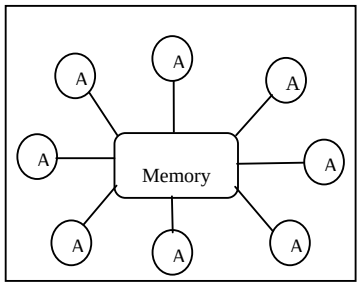

Figure2. An instance of ATeams architecture.

ATeams thus sit in the intersection between a numbers of different problem solving methodologies. In particular, they offer a convenient architecture for implementing hybrid systems. They can support flexible distributed computing. Finally, they allow existing algorithms to be reused (with some limited modification). ATeams thus promise an efficient framework for building combinatorial optimization systems [4].

3. PROPOSED MULTI AGENT SYSTEM

We propose a multi agent system that consists of initial scheduling agent, search agents, and schedule management agent. The architecture of our multi agent system is shown in Figure3. Initial scheduling agent generates a good solution for the job shop problem by a heuristic approach. Search agents individually improve the solution of problem by iterative approaches. Scheduling management agent is a common memory that saves the best solution of search agents and is an interface between search agents.

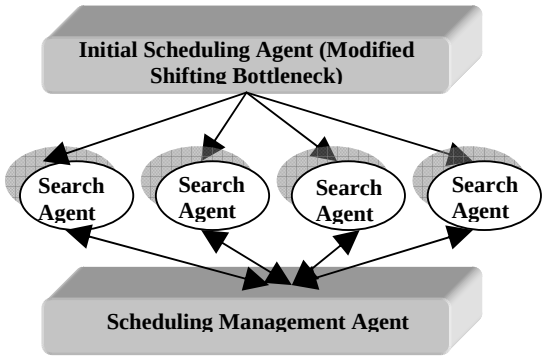

Figure3. Framework of proposed multi agent system.

The architecture has been implemented using client – server programming. First, initial scheduling agent starts to generate feasible solution and send the solution and related information to the search agents. Once search agents receive the information from initial scheduling agent, they individually improve the solution. Each of search agents is implemented as a separate client, which connects to the server. Scheduling management agent is similar to common memory and is server process. The server maintains the best solution and related information such as makespan and sequence of operations. When the client (search agent) is connected to the server (scheduling management agent), depending on the instantiation of the server, an agent may replace the solution it read, or it may add the new solution. This process continues until the termination criterion is satisfied. Termination criterion is to obtain the optimum solution (lower or upper bound for unknown optimum solution) or to carry out the number of evaluation.

In this approach, the cooperation between agents is possible because one agent can work on the output of another. Thus it may increase synergy between agents and obtain the better solutions for the problem rather than individual agents. Also this approach allows parallel computations that may decrease the computational efforts for the problem.

A. Modified Shifting Bottleneck

One of the components of the proposed multi agent system is initial scheduling agent. We can use any heuristics presented in the literature for this agent. Among them, a successful approach is the Shifting Bottleneck (SB). This paper presents a modified Shifting Bottleneck that is an effective heuristic approach which can generate a good solution in a low computational effort. In this section, the details of this approach are explained.

SB was, first, presented by Adams in [1].Later the other researchers have enhanced this method [10][3][5][22]. There have been extensive computational experiments evaluating the performance of several versions of SB routine on different shop configurations [12],[30]. The general consensus is that the SB performs quite well compared to various dispatching rules on almost all problem types.

The main strategy of the SB lies in relaxing the problem into m single machine subproblems and solving each single machine independently. This approach consists of four functions: problem decomposition, bottleneck identification, subproblem scheduling, and reoptimization. On a specified scheduling criterion, the machine having the maximum lower bound is selected as the bottleneck machine and the SB sequences the bottleneck machine first while ignoring the remaining unscheduled machines. After the machine is scheduled, the reoptimization procedure is triggered. The SB algorithm repeats the single machine scheduling procedure until all machines are scheduled [34].

Subproblems solution procedures (SSPs) and reoptimization are two main functions in SB [12], [30].

They found that the better SSPs and reoptimization has higher solution quality and a system with more significant bottleneck machines might have higher search efficiency. In the large scale problems, implementing heuristic approaches for subproblems solution procedure in SB can decrease computational efforts. Although using the exact approaches for single machine (subproblems) might improve solution quality of the job shop problems, this usually increases computational effort in large scale problems. Reoptimization is important problem in SB that can increase computational efforts especially in the large scale problems. This paper presents a new approach that can omit reoptimization.

I) Subproblems Solution Procedure (Single Machine)

Schrage algorithm, a heuristic method suggested by Schrage in [25], is an effective algorithm that is used in single machine problem. The algorithms of McMahon and Florian and Carlier are based on Schrage algorithm [21], [9]. In both methods Schrage heuristic is used at every node of a search tree to generate a complete solution. Thus, a good solution of Schrage heuristic can decrease the space of search in the enumeration approaches such as Carlier algorithm. Also computational efforts of heuristic approaches such as Schrage algorithm is lower than the exact algorithms such as branch and bound (Carlier algorithm in [9]) in single machine problem. This can lead to decrease computational efforts in the job shop problems. Wenqi and Aihua presented a heuristic as Schrage algorithm with disturbance for solving subproblems [32].Then, they base on the heuristic, presented improved shifting Bottleneck (ISB) and showed that it has a better performance than SB [32]. However, it has some drawbacks. This paper tries to resolve some of the drawbacks and present a modified Schrage algorithm that is more effective than previous Schrage algorithm and Schrage algorithm with a disturbance (DS).

In a single machine problem with heads and tails, n independent jobs should be sequenced on a machine: a job i is available for processing by the machine at time r_i, has to spend an amount of time p_i on the machine and an amount of time q_i in the system after its processing by the machine. The objective is to minimize the makespan.

Carlier shows a sequence, in this problem, with a conjunctive graph $G=(X, U)$ [9]. The set X of nodes is obtained by adding two nodes O and * to the set i of jobs: $X= I \bigcup \{O,*\}$, where, O is a job 'beginning', and * a job 'end'. The set U of arcs includes three sets: $U=U_1 \bigcup U_2 \bigcup U_3$. Let $U_1 = \{(O,i)| \ i \in I\}$; arc(O,i) is valued by r_i so that job i cannot start before the point in time r_i. Let $U2 = \{(i,*)| \ i \in I\}$; arc $(i,*)$ is valued by $q_i + p_i$ since job i has to spend an amount of time $q_i + p_i$ in the system after its beginning of processing by the machine. Let $U3$ = {(i,j)| job i precedes job j in the sequence}; arc(i,j) is valued by p_i; these arcs set the sequence. The aim is to find a sequence that minimizes the value of the critical path in the associated conjunctive graph.

In the Schrage algorithm the job ready with greatest q_i is scheduled first. In this algorithm, U is the set of jobs already scheduled and $\overline{U}$ is the set of jobs to be scheduled, and t is the time. In Schrage algorithm, when $r_i<r_j$ and $q_i>q_j$ the algorithm can generate the optimal solution and when $r_i<r_j$ and $q_i<q_j$ it may result in weak solution [9]and [32]. According the Schrage algorithm, in each stage, only the ready jobs, ($r_i <= t$) from set $\overline{U}$, are sequenced. In the following we denote these jobs by the set R. However, it is possible that there exists a job k ($k \in \overline{U}$) so that $r_k>t$ while the tail of job k (i.e., q_k) is very larger than the tail of job i ($\forall \ i \in R$). In this situation it is logically better to take into account job k in the set R. This problem may lead to increase makespan. Therefore, we need to expand the scope of the set R to all jobs in $\overline{U}$ so that the jobs with large tail that are not ready have chance to be selected. But the main problem rises which jobs can be added? In other words, in Schrage algorithm, if ri < rj and qi > qj , i is earlier sequenced j that is logically true and proved in literature [9]. But if ri < rj and qi < qj we deal with this challenge that which of them must earlier be sequenced. This problem is studied by the following theorem.

Theorem 1. In one machine sequencing problem, if there are two jobs i and j with properties $0 \leq$ ri<rj and qi<qj, a necessary condition for sequencing j before i is (qj>m+ qi & pi > m) where , rj-t=m and pi >=0.

Proof. We demonstrate the sequence of jobs with a conjunctive graph G=(X, U) similar to Carlier algorithm (Figure4).

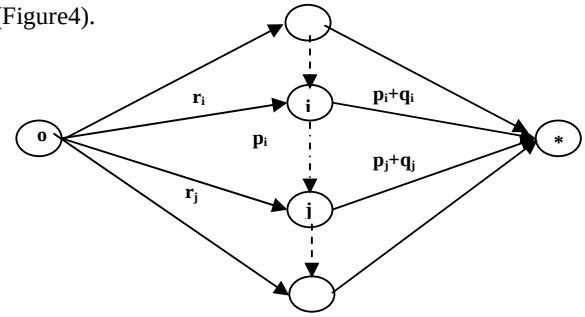

Figure4. Associated conjunctive graph.

Let U be the set of jobs already scheduled, $\overline{U}$ be the set of all other jobs and ck be the completion time of the jobs belonging to set U in stage k. job i is a job with minimum r in the set $\overline{U}$. Let t be max (ck, ri). Assume that the Job j is a job with rj so that rj > t and qj > qi. We prove that a necessary condition for sequencing j before i is: q j> m+qi & pi > m, where, rj-t=m and pi >=0. Here, there exist two options as follows:

Option 1: job i to be scheduled before job j

Let Li1 be the length of path that pass through 0, 1, i,

* and Lj1 be the length of path that pass through 0, 1, j, * and L1=Max (Li1, Lj1). Thus we have:

L1=Max (t+pi+qi , t+pi+pj+qj) (1)

Option 2: job j to be scheduled before job i

Let L_i^2 be the length of path that pass through 0, 1, *i*, * and L_j^2 be the length of path that pass through 0, 1, *j*, * and $L_2=Max\ (L_i^2, L_j^2)$. Then,

$L_2=Max\ (r_j+p_j+q_j\ ,\ r_j+p_j+p_i+q_i)$ (2)

Assume that job j is sequenced earlier than job i thus L2 must be lower than L1 because the aim is to minimize the longest path through 0 to *. Therefore, the formula (3) should be true.

$Max\ (r_j+p_j+q_j\ ,r_j+p_j+p_i+q_i)<Max(t+p_i+q_i\ ,t+p_i+p_j+q_j)$ (3)

This means the formulae (4) or (5) should be true:

$Max(r_j+p_j+q_j\ ,\ r_j+p_j+p_i+q_i)< t+p_i+q_i$ (4)

$Max(r_j+p_j+q_j\ ,\ r_j+p_j+p_i+q_i)< t+p_i+p_j+q_j$ (5)

The formula (4) is true if the formulae (6) and (7) both are true.

$r_j+p_j+q_j < t+p_i+q_i$ (6)

$r_j+p_j+p_i+q_i< t+p_i+q_i$ (7)

The formula (5) is true if the formulae (8) and (9) both are true.

$r_j+p_j+q_j < t+p_i+p_j+q_j$ (8)

$r_j+p_j+p_i+q_i < t+p_i+p_j+q_j$ (9)

Let $r_j-t=m$ then : $r_j=m+t$ (10)

By replacing (10) in the formulae (6),(7),(8),(9), respectively we have:

$m+t+p_j+q_j < t+p_i+q_i \Rightarrow q_j < q_i + p_i - p_j - m$ (11)

$m+t+p_j+p_i+q_i< t+p_i+q_i \Rightarrow m+p_j<0$ (12)

$m+t+p_j+q_j < t+p_i+p_j+q_j \Rightarrow p_j>m$ (13)

$m+t+p_j+p_i+q_i < t+p_i+p_j+q_j \Rightarrow q_j>q_i+m$ (14)

By the assumption: pj >=0 and m >=0, thus, Formula (12) is false. That means formula (7) is false, then the Formula (4) is false all the times. Formulae (10) and (11) by the assumption are true. That means Formulae (8) and (9) are true. So Formula (5) is also true all the times. Thus, Formula (3) is true that means L2<L1 and the theorem is proven.

If the job *j* to be critical, i.e., belonging to critical path then as mentioned above necessary condition for sequencing j before i is (qj > m + qi & pi > m). But if the jobs j is not critical then sequencing it before i can be lead to increase delay (rj - t) and makespan. We define historically a condition for criticality of the job j by following.

$t+p_i+p_j+q_j>h(s);\ h(s)=t_s+\sum_{k\in\overline{U}_i}p_k+\underset{k\in\overline{U}_i}{Min}\,q_k$ (15)

Proposition1. In stage *s* and $k\in\overline{U}$ $h(s)=t_s+\sum_{k\in\overline{U}_i}p_k+\underset{k\in\overline{U}_i}{Min}\,q_k$, is a lower bound on the optimal makespan.

Proof. Carlier [13] proved that for all $I_1\subseteq I$, $h(I_i)=\underset{i\in I_i}{Min}\,r_i+\sum_{i\in I_i}p_i+\underset{i\in I_i}{Min}\,q_i$ is a lower bound on the optimal makespan. Since $\overline{U}\subseteq I$, therefore h(s) is a lower bound on the optimal makespan.

Now we present a modified Schrage algorithm according to theorem1. The steps of algorithm are shown in Figure 5. In this algorithm, in each stage, both ready jobs and the jobs with large tails (based on theorem1) can be sequenced. This improves the result of sequencing. Also in spit of DS algorithm, all of the unscheduled jobs are not candidate and are limited by the necessary condition. This can decrease computational effort in each stage especially in large problems. Furthermore, this algorithm does not depend on any coefficient such as δ. Experimental results show that M.A.S gets much better solutions than DS and SA.

Step1: Let $t=\underset{i\in I_i}{Min}\,r_i$, $p=p_l$; $U=\Phi$, *l* is index of $\underset{i\in I_i}{Min}\,r_i$

Step2: Find $r_i <= t+p$; $i\in\overline{U}$, If $r_i <= t$, then $u_i = q_i$, elseif $r_i > t$ & $t+p+p_i+q_i> t+\sum_{k\in\overline{U}_i}p_k+\underset{k\in\overline{U}_i}{Min}\,q_k$, $k\in\overline{U}$ then $u_i = q_i-(r_i - t)$; otherwise $u_i = 0$.

Step3: Choose job with the greatest u_i (If there are ties, break them by giving preference to the minimum r_i. If there are still ties, break them by giving preference to the greatest q_j and are still ties, break them by giving preference to the greatest p_j) , Set $t_j = max\{t; r_j\}$, $U=U\bigcup\{j\}$.

Step4: Set $t = max\{tj+p_j; \underset{i\in\overline{U}}{Min}\,r_i\}$. If $U = I$, stop; otherwise, return to Step 2.

Figure5. Modified Schrage algorithm

II) Modified Shifting Bottleneck

This paper also presents a new approach for omitting reoptimization in SB procedure. In the proposed approach, the ready operations are sequenced first while the operations on critical machine should be preferenced. This can omit the reoptimization efforts. In our approach similar to SB, the job shop problem is decomposed into m single machines, but by the deferent way. In this approach, the ready operations on the same machine lie in a block and each of them is a single machine problem. In each stage, the operations in a block should be sequenced by a subproblem solution procedure, while the other unscheduled operations on the machine should be considered in the problem. If the operation on the block has not been prioritized, it might have delayed.

For the sake of simplicity, we define critical machine in each stage as follows:

$$m^* = \max_{j \in M}(\min_{o_{ij} \in \overline{U}} r_{ij} + \sum_{o_{ij} \in \overline{U}} p_{ij}) \; ;j=1,2...m,i=1,2... \quad (16)$$

n

Subproblem solution procedure is based on modified Schrage algorithm presented in the pervious section. In this procedure, the prioritization is based on the head and tail of the operations (u_{ij}), however as stated above, we should prioritize the operations of critical machines, where, criticality can be considered as a fuzzy concept. All of the machine can be critical by a membership degree. Membership degree of a machine can be calculated as follows:

$$\mu_j = \frac{\min_{o_{ij} \in \overline{U}} r_{ij} + \sum_{o_{ij} \in \overline{U}} p_{ij}}{\max_{j \in M}(\min_{o_{ij} \in \overline{U}} r_{ij} + \sum_{o_{ij} \in \overline{U}} p_{ij})}; \; j = 1,2 \ldots m \, , i = 1,2 \ldots . n \quad (17)$$

The delay on critical machine can increase makespan but in no critical machine the operations can be delayed. We use this concept in our approach. In the proposed approach, the criticality of machines takes into account in subproblem solution procedure. Subproblem solution procedure is defined in Figure6.

Step1: Let t = $\underset{o_{ij} \in I_j}{Min} r_{ij}$,

Step2: p = p_l; l is index of $\underset{o_{ij} \in I_j}{Min} r_{ij}$,Find r_{ij}<= t+p; $o_{ij} \in I_j$,

If $o_{ij} \in V_j$, y_1=.05, otherwise y_1=0

If $r_{ij} > t$ and $\mu_j > .97$ then y_1=.05, otherwise y_1=0

If $r_{ij} > t$ and $p+p_{ij}+q_{ij} \succ \sum_{o_{ij} \in I_j} p_{ij} + \underset{o_{ij} \in I_j}{Min} q_{ij}$

Then $y_3 = 1.5 \times (r_i - t)$

If $r_{ij} > t$ and $p+p_{ij}+q_{ij} <= \sum_{o_{ij} \in I_j} p_{ij} + \underset{o_{ij} \in I_j}{Min} q_{ij}$

Then $y_3 = 3 \times (r_i - t)$,Otherwise , y_1=0

Step3: set $u_{ij} = (q_{ij} - y_3) * (1 - y_1 - y_2)$

Step4: Choose operation with the greatest u_i (If there are ties, break them by giving preference to the minimum r_i. If there are still ties, break them by giving preference to the greatest q_j and are still ties, break them by giving preference to the greatest p_j) ,

If $o_{ij}^* \in V_j$ Set $st_{ij} = max\{t; r_{ij}\}$, $I_j = I_j \bigcup \{o_{ij}\}$, $U = U \bigcup \{o_{ij}\}$, o_{ij}^* is labeled in set V_j otherwise, stop.

Step5: Set $t = max\{ st_{ij} + p_j; \underset{o_{ij} \in I_j}{Min} r_{ij} \}$. If all operations of set V_j are labeled , stop; Otherwise: return to Step2.

Figure6. Subproblem solution procedure

The symbols used in this algorithm are as follows:

N: number of jobs *M: number of machine*
j_i: *job i* m_j: *Machine j*
o_{ij}: *Operation related to job i on machine j*
U: set of scheduled operations
r_{ij}: *the release time(head) of operation* o_{ij}
q_{ij}.: *the delivery time(tail)of operation* o_i
p_{ij}: *the processing time(tail)of operation* o_{ij}
V_j *: set of the ready operations on machine j*
I_j *: set of unscheduled operations on machine j*
S_{ij}: *set of the successor operations of operation* o_{ij}

The proposed algorithm for the job shop problem based on shifting Bottleneck is shown in Figure7.

Step1: Let $U = \Phi$
Step2: calculate r_{ij}, q_{ij}.
Step3: find critical machine (*m**).
Step4: find the ready operations of set $\overline{U}$ *(r_{ij}=0),* cluster the ready operations in same machine and find V_j.
Step5: if $V_j \neq \Phi$ sequence the operations of set V_j by the subproblem solution procedure, $\forall j \in M$.
Step6: if $\overline{U} = \Phi$, stop; otherwise, return to step 2.

Figure7. Modified Shifting Bottleneck.

B. Search Agents

Search agents are important components of our multi agent systems. There are many intelligent search procedures or meta-heuristics such as genetic algorithm (GA)[13], simulated annealing (SA), tabu search (TS) and guided local search, etc [13][20][11][24][6]. These approaches may take into account a search agent. The stated approaches are based on the local search principle where optimization starts from a given initial solution and iteratively generates new solutions, each of which is obtained from the previous one by performing a move (a small perturbation) on it. The set of allowed moves is specified by a neighborhood function, which is defined for every feasible schedule [18]. We, in this paper, present a hybrid search approach for each agent.

A neighborhood structure is a mechanism which can obtain a new set of neighbor solutions by applying a small perturbation to a given solution. Each neighbor solution is reached immediately from a given solution by a move [17]. Neighborhood structure is directly effective on the efficiency of search algorithm. Therefore, unnecessary and infeasible moves must be eliminated if it is possible. The first successful neighborhood structure for the JSP was introduced by Van Laarhoven in [31], and is often denoted by N1 [7]. The N1 neighborhood is generated by swapping any adjacent pair of critical operations on the same machine.

However, the size of the neighborhood N1 is quite large and includes a great number of unimproved moves. An important observation is that unless the job-

predecessor of u or the job-successor of v is on the critical path P(0, n), the interchange containing u and v cannot reduce the makespan, i.e., swapping internal operations within a block never gives an immediate improvement on the makespan [20]. Therefore, the further refined neighborhoods N4, N5, and N6 have been proposed by [11][24][6]. Those recently well-known neighborhoods are mainly based on the concept of block, in which a move is defined by inserting an operation to either the front or the rear of the critical block. The neighborhood N5 involves the reversal of a single border arc of a critical block3 and is substantially smaller than the other neighborhoods, whereas the neighborhoods N4 and N6 involve the reversal of more than one disjunctive arc at a time and thus could investigate a considerably larger neighborhood. The neighborhood N6, which is also considered as an extension of the neighborhood N5, is more constrained than the neighborhood N4 and is currently one of the most effective and efficient neighborhood structures.

A key component of the above neighborhood structures is the critical path, which is the longest route from start to end in directed graph Ds = (V, $A \cup S$) and whose length represents the makespan. Any operation on the critical path is called a critical operation. It is also possible to decompose the critical path into a number of blocks. A block is a maximal sequence of adjacent critical operations that is processed on the same machine.

We use each of the neighborhood structures in each agent. But the basic problem in local search is that how to avoid being trapped at a local optimum. We use another neighborhood function as exchange function, in order to avoid being trapped at a local optimum. Exchange is a neighborhood function used to move around in which any two randomly selected operations are simply swapped [26]. Furthermore, we can apply the reoptimization phase in Shifting Bottleneck approach for improving the solution obtained.

The proposed search agent combines the neighborhood search, random search and reoptimization phase in Shifting Bottleneck. The procedure starts with a feasible initial solution and stores it as the current seed and the best solution. The initial solution is generated by initial scheduling agent (modified shifting Bottleneck).The neighbors of the current seed are then produced by a neighborhood structure N1 to N6 [35]. These are candidate solutions. They are evaluated by an objective function, and a candidate which is the best or satisfies the aspiration criterion is selected as new seed solution. This selection is called a move. If the new seed solution is better than the current best solution, it is stored as new best solution. If the candidate solutions do not satisfy the aspiration criterion then generate neighbors of the current seed solution by Exchange function. Next, the reoptimization phase of MSB is executed on the current seed and found the best solution as current seed. Iterations are repeated until a stop criterion is satisfied. Figure12 shows the proposed search algorithm. The configuration of the above procedure is shown in Fig.8.

4. EXPERIMENTAL RESULTS

We applied the proposed multi agent system to solve some 10×10 benchmark problems, specifically FT10, ABZ5, ABZ6, LA16, LA17, LA18, LA19, and LA20 [15][1][19].A 10×10 problem has 10 jobs and 10 machines, and thus 100 variables

The proposed multi agent system approach (MAS) is compared by single agent. Comparisons are based on three main factors: Mean of makespan (Me.), Standard Deviation, Best solution (B.S). CPU time and evaluations as computations efforts is fixed for two approaches. The results for ten experiments are represented in table1.

Experiment results show the superiority of the proposed MAS approach in comparison to single agent. As shown in table 2, the mean of makespan and best solution in the proposed approach are lower than single agent approach. Standard deviation is almost same for two approaches. Thus the proposed approach generated the better results. The results have obtained in equal evaluations.

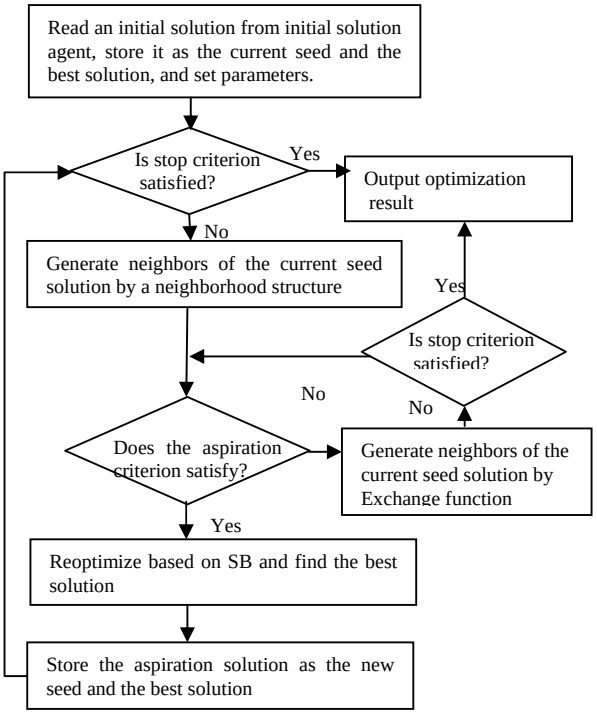

Figure8. Flowchart of proposed search algorithm.

5. CONCLUSIONS AND FUTURE WORKS

We present a multi agent system for the job shop

scheduling problems. The proposed system consists of initial scheduling agent, search agents, and schedule management agent. In initial scheduling agent, a modified Shifting Bottleneck is proposed. That is, an effective heuristic approach and can generate a good solution in a low computational effort. In search agents, a hybrid search approach is presented. The schedule management agent can manage the system. Finally, the proposed agent based system is tested and validated by some benchmark problems. The results show the superiority of the proposed system in terms makespan minimization and CPU times.

We implemented this approach for the classic job shop problem. In future research, we can apply for the other problems in scheduling. Moreover, respond to this question that how much the number of agents and communication frequencies effect on solution quality and computations efforts, can be future researches.

TABLE 1 COMPARISON OF THE PROPOSED APPROACH TO THE SINGLE AGENT FOR INSTANCE PROBLEMS

problem	O.N	Single agent			M.A.S proposed		
		Me.	B.S	SD	Me.	B.S	SD
FT10	930	1028.7	1005	10.66	1014.3	985	13.53
ABZ5	1234	1278.6	1250	11.77	1271	1249	12.95
ABZ6	943	951.7	948	4.29	949.5	948	1.58
LA16	945	1025.1	990	15.76	1010.5	988	16.33
LA17	784	806.7	793	14.02	799.7	792	11.50
LA18	848	873.9	853	14.23	867.3	853	11.54
LA19	842	876.6	875	1.45	876.2	875	1.03
LA20	902	934.9	918	9.12	927.6	918	7.65

6. REFERENCES

[1] J. Adams, E. Balas, D. Zawack, "The shifting bottleneck procedure for job shop scheduling," Management Science, 34, 391–401, 1988.

[2] K. Anke, R Staudte and W. Dilger, Producing and improving timetables by means of constraints and agents, Technical Report WS-97-05, AAAI Press, Menlo Park, CA,pp. 142-147, 1997.

[3] D. Applegate, W. Cook, "A computational study of job shop scheduling problem," ORSA Journal of Computing, 3149–156, 1991.

[4] M. E. Aydin, T.C. Fogarty, Teams of autonomous agents for job-shop scheduling problems: An Experimental Study, Journal of Intelligent Manufacturing, 15(4), (2004) 455–462.

[5] E. Balas, J.K. Lenstra, Vazacopoulos, "The One machine Problem with Delayed Precedence Constraints and its use in Job Shop Scheduling," Management Science, 41, 1, 94-109, 1995.

[6] E. Balas, A.Vazacopoulos, Guided local search with shifting bottleneck for job shop scheduling. Management Science, 44(2):262–75, 1998.

[7] J Blazewicz, W Domschke, E. Pesch, The job shop scheduling problem: conventional and new solution techniques. European Journal of Operational Research, 93, 1–33, 1996.

[8] P. Burke and P. Prosser, The Distributed Asynchronous Scheduler, in: Intelligent Scheduling, M.B. Morgan, ed., Morgan Kaufman Publishers, Inc., San Francisco, pp. 309–339, 1994.

[9] J. Carlier, "The one-machine sequencing problem," European Journal of Operational Research, 11, 42-47, 1982.

[10] S. Dauzere-Peres, J.B. Lasserre," A modified shifting bottleneck procedure for job-shop scheduling," International Journal of Production Research, 31, 923-932, 1993.

[11] M. Dell'Amico, Trubian M. Applying tabu-search to job-shop scheduling problem.Annals of Operations Research, 41,231–52, 1993.

[12] E. Demirkol, S.V. Mehta, R. Uzsoy," A computational study of shifting bottleneck procedures for shop scheduling, " J. Heuristics, 3, 111–137, 1997.

[13] U. Dorndorf, E. Pesch, Evolution based learning in a job shop scheduling environment. Comput. Oper. Res., 22, 25–44, 1995.

[14] J. Ferber, (1999) Multi-agent systems: An introduction to Distributed Artificial Intelligence. Addison Wesley, London.

[15] H. Fisher and D.L. Thompson, "Probabilistic learning combinations of local job shop scheduling rules," In J. F. Muth, G. L. Thompson (eds.), Industrial Scheduling, Prentice-Hall, Englewood Cliffs, 1963.

[16] M.R. Garey, D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP Completeness," San Francisco Freeman, 1979.

[17] F. Glover, M. Laguna, Tabu search. Dordrecht: Kluwer Academic Publishers; 1997.

[18] AS. Jain, S. Meeran, Deterministic job shop scheduling: past, present and future. European Journal of Operational Research, 113, 390–434, 1999.

[19] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, "Recent developments in deterministic sequencing and scheduling: a survey" in M. A. H. Dempster, J.K. Lenstra and A.H.G. Rinnooy Kan , Deterministic and Stochastic Scheduling, Reidel, Dordrecht, 1982.

[20] HD Matsuo, CJ Suh, RS Sullivan, A controlled search simulated annealing method for general job shop scheduling problem. Working Paper, 03-04-88, Department of Management, The University of Texas at Austin, Austin, TX, 1988.

[21] G.B. McMahon and M. Florian, "On scheduling with ready times and due dates to minimize maximum lateness," Operations Research, 23, 415-482, 1975.

[22] S. Mukherjee and A.K. Chatterjee, "On the representation of the one machine sequencing problem, " Eur. J. Oper. Res, doi:10.1016/j.ejor.2006.07.024.

[23] S. Murthy, "Synergy in Cooperating Agents: Designing Manipulators from Task Specifications," Ph.D. dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, 1992.

[24] E. Nowicki, C. Smutnicki, A fast taboo search algorithm for the job shop scheduling problem.Management Science, 42, 6, 797–813, 1996.

[25] L. Schrage, "Obtaining optimal solutions to resource constrained network scheduling problem," Unpublished manuscript 1971.

[26] M Sevkli, M. E. Aydin, Variable Neighbourhood Search for Job Shop Scheduling Problems, Journal of software, V.1, N. 2, August 2006.

[27] W. Shen, Q. Hao, H. J. Yoon, D.H. Norrie, Applications of agent-based systems in intelligent manufacturing:An updated review, Advanced Engineering Informatics 20 (2006) 415–431.

[28] S.Talukdar, Asynchronous teams. Proceedings of the 4th international symposium on expert systems applications to power systems, LaTrobe university,Melbourne, Australia.
[29] A. Tharumarajah and R. Bemelman, Approaches and issues in scheduling a distributed shop floor environment, Computers in industry, 34, 95-109, 1997.
[30] R. Uzsoy and C.S. Wang, "Performance of decomposition procedures for job shop scheduling problems with bottleneck machines," Int. J. Prod. Res., 38, 1271–1286, 2000.
[31] PJM. Van Laarhoven, EHL Aarts, JK. Lenstra, Job shop scheduling by simulated annealing. Operations Research , 40, 1, 113–125, 1992.
[32] H. Wenqi and Y. Aihua, "An improved shifting bottleneck procedure for the job shop scheduling problem," Computer & Operations Research, 31, pp. 2093-2110, 2004.
[33] M. Wooldridge, An introduction to multi-agent systems. John Wiley & Sons, Ltd., Chichester, England. , 2002.
[34] C.S. Wu, D.C. Li, T.I. Tsai, "Applying the fuzzy ranking method to the shifting bottleneck procedure to solve scheduling problems of uncertainty," Int. J. Adv. Manuf. Technol., 31, 98–106, 2006.
[35] C.Y. Zhang, P. Li, Z. Guan, Y. Rao, A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem Computers & Operations Research doi:10.1016/j.cor.2005.12.002.
[36] Z. D. Zhou, H. H. Wang, Y. P. Chen, W. Ai, S. K. Ong, J. Y. H. Fuh and A. Y. C. Nee, A Multi-Agent-Based Agile Scheduling Model for a Virtual Manufacturing Environment, Int. J Adv Manuf. Technol., 21, 980–984, 2003.