Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (6): 992-999, 2008
ISSN 1812-5654
© 2008 Asian Network for Scientific Information

A Heuristic Approach for Large Scale Job Shop Scheduling Problems

M.H. Karimi Gavareshki and M.H. Fazel Zarandi
Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran

Abstract: This study presents a heuristic approach based on Shifting Bottleneck (SB) for large scale job shop scheduling problems. 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 study, first, 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 algorithm is tested and validated. Experiment results show the superiority of our approach in comparison to SB in large scale problems especially in computational efforts. This approach can be a good initial seed in using search techniques.

Key words: Job shop, Schrage algorithm, scheduling, shifting bottleneck, single machine

INTRODUCTION

The job-shop problem is regarded as one of the hardest in combinatorial optimization (Brinkkotter and Brucker, 2001). During the past decades, many researchers have been focusing on the job shops and proposed several effective algorithms for them. They can be classified as optimization and approximation algorithms. The optimization algorithms are usually based on the branch and bound scheme (Brucker *et al*., 1994; Carlier and Pinson, 1994). These algorithms have made considerable achievement; however, their implementation needs high computational cost. On the other hand, approximation algorithms are more effective for large size problem instances. Among the heuristics, a successful approach is the Shifting Bottleneck (SB).

SB was, first, presented by Adams *et al.* (1988). Later Dauzere-Peres and Lasserre (1993), Applegate and Cook (1991), Balas *et al.* (1995) and Mukherjee and Chatterjee (2006) have enhanced this method. There have been extensive computational experiments evaluating the performance of several versions of SB routine on different shop configurations (Demirkol *et al*., 1997; Holtsclaw and Uzsoy, 1996; Uzsoy and Wang, 2000). 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 (Wu *et al*., 2006).

Subproblems solution procedures (SSPs) that are single machine problems and reoptimization are two main functions in SB. Holtsclaw and Uzsoy (1996) tested the quality and efficiency of the solution on 12 combinations of machine criticality measures that used for selecting shifting machines and subproblem solution procedures (SSPs). According to their research, the more sophisticated algorithm usually has the better efficiency. Demirkol *et al*. (1997) found that the better SSPs and reoptimization has higher solution quality. Uzsoy and Wang (2000) modified the testing structure of Demirkol *et al*. (1997) to distinguish the bottleneck machines from non-bottleneck. They found that 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 such as Carlier algorithm (1982) for single machine (subproblems) can improve solution quality of

Corresponding Author: M.H. Fazel Zarandi, Department of Industrial Engineering, Amirkabir University of Technology, P.O. Box 15875-4413, Tehran, Iran Tel:+98(21)64542786 Fax:+98(21)66413969

the job shop problems, this increases computational effort in large scale problems. Wenqi and Aihua (2004) presented a heuristic as Schrage algorithm with disturbance for solving subproblems, base on presented improved shifting bottleneck (ISB) and showed that it has a better performance than SB. However, it has some drawbacks. This study 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).

Reoptimization is one of the important problems in SB that can increase computational efforts especially in the large scale problems. This study presents a new approach that can omit reoptimization.

Job shop problem: The job shop scheduling problem can be described as follows: There are a set of jobs and a set of machines. Each job consists of a series of operations and each operation with fixed processing time is processed by a certain machine. The problem consists in scheduling the jobs on the machines with the objective to minimize the makespan the time needed to finish all the jobs. 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.

Let N = {0, 1,..., n} denotes the set of operations (with 0 and n the dummy operations .start. and .finish.), M the set of machines, A the set of pairs of operations constrained by precedence relations and E_k the set of pairs of operations to be performed on machine k and which therefore cannot overlap in time. Further, let p_i denote the (fixed) duration (processing time) and t_i the (variable) start time of operation i. The job shop problem can then be stated as:

$$\begin{aligned} &\min t_n \\ &t_j - t_i >= p_i, \quad (i,j) \in A, \\ &t_i >= 0, \quad i \in N, \\ &t_j - t_i >= p_i \vee t_i - t_j >= p_j, \ (i,j) \in E_k, k \in M. \quad (P) \end{aligned}$$

Any feasible solution to (P) is called a schedule. It is useful to represent this problem on a disjunctive graph G = (N, A, E), with node set N, ordinary (conjunctive) arc set A and disjunctive arc set E (Balas, 1969). Conjunctive arcs represent precedence relations and pairs of disjunctive arcs correspond to operations that must be processed on the same machine. Figure 1 shows the graph for a problem with 8 operations (on three jobs) and three machines.

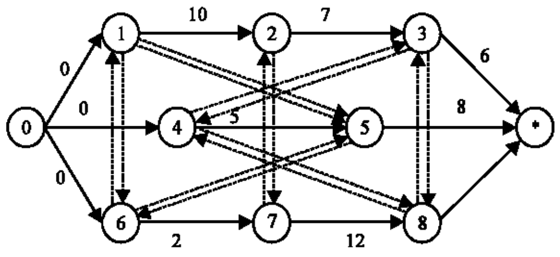

Fig. 1: Digraphy an instance

A selection S_k in E_k contains exactly one member of each disjunctive arc pair of E_k. Actually, determining an acyclic section on S_k is equivalent to sequencing the jobs (operations) on machines k. A complete selection S consists of the union of selection S_k, one in each E_k, k ∈M. In the language of disjunctive graphs, the problem is to find an acyclic complete selection S ⊂ E that minimizes the length of a longest path in the directed graph D_S.

Shifting bottleneck: Shifting Bottleneck is a heuristic approach that uses in job shop scheduling problems. 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 (Wu *et al*., 2006).

Let M_0 be the set of machines that have already been sequenced, by choosing selections Sp (p∈M_0). Let (P(k, M_0)) be the problem obtained by replacing each arc set Ep (p∈M_0) with the corresponding selection Sp and deleting each arc set Ep (p∈M/M0-{k}). A bottleneck machine m∈M/M0 is such that v(m,M0) = max{v(k,M0): k∈M/M0}, where v(k,M0) is the value of a good solution to (P(k,M0)) Fig. 2 (Wenqi and Aihua, 2004; Adams *et al*., 1988).

Let M_0 be the set of already sequenced (M_0 = Φ at the start).

In step 1, to identify the next bottleneck machine to be sequenced, for each k∈M/M_0 the following problem should be solved:

$$\begin{aligned} &\min t_n \\ &t_j - t_i >= p_i, \quad (i,j) \in \cup(S_p : p \in M_0) \cup A, \\ &t_i >= 0, \quad i \in N, \\ &t_j - t_i >= p_i \vee t_i - t_j >= p_j, \ (i,j) \in E_k, k \in M. \quad (P (k, M0)) \end{aligned}$$

Step 1: Identify a bottleneck machine m among the machine $k \in M/M_0$ and sequence it optimally. Set $M_0 \leftarrow M_0 \cup \{m\}$.
Step 2: Reoptimize the sequence of each critical machine $k \in M$, while keeping the other sequences fixed; i.e., set $M'_0 := M_0 - \{k\}$ and solve P (k, M′ 0). Then if M_0 is equal to M, the algorithm stops; otherwise go to step 1.

Fig. 2: Shifting bottleneck procedure

Also, to reoptimize in Step 2 the sequence of each critical machine $k \in M_0$, for each of such machines, a problem of the form (P (k, M'_0)) is solved for some subset $M'_0 \subset M_0$.

The problem (P(k,M0)) is equivalent to that of finding a sequence for machine k that minimizes the maximum lateness, given that each operation i to be performed on machine k has, besides the processing time P_i, a release time r_i and a due date d_i. Here, $r_i = L(0, i)$ and $d_i = L(0, n)$ $L(i,n) + p_i$, with L(i, j) the length of a longest path from i to j in D_t and T: $= \cup(S_p: p \in M_0)$. This latter problem in turn can be viewed as a minimum makespan problem where each job has to be processed in order by three machines, of which the first and the third have infinite capacity, while the second one (corresponding to machine k in the above model) processes one job at a time and where the processing time of job i is r_i on the first machine, p_i on the second and $q_i : = L(0,n)\ d_i$ on the third machine. The numbers r_i and q_i are sometimes referred to as the head and tail of job i (Wenqi and Aihua, 2004; Adams *et al.*, 1988).

Thus, the one-machine problems solved during the algorithm are of the following form:

$$\begin{aligned} &\min t_n \\ &t_n - t_i >= p_i + q_i, \quad i \in N^*, \\ &t_i >= r_i, \quad i \in N^*, \\ &t_j - t_i >= p_i \vee t_i - t_j >= p_j,\ (i,j) \in E_k,\ k \in M - M_0 \qquad (P^*(k, M_0)) \end{aligned}$$

Where, r_i and q_i are defined as above and N* is the set of jobs to be processed on machine k.

The problem (P*(k, M_0)), is a single machine problem with head and tail that is denoted the problem $1|r_j, q_j|C\max$ in the literature. Further we present a heuristic for solving this problem.

A new heuristic for single machine: It is shown that the problem $1|r_j, q_j|C\max$ is strongly NP-hard. A number of special cases of this problem are solvable in polynomial time (Vakhania, 2004). The enumerative methods for solving $1|r_j, q_j|C\max$ are studied by Baker and Su (1974), McMahon and Florian (1975), Carlier (1982) and Grabowski *et al.* (1986). The algorithm proposed by Carlier was successfully tested even for 10000 jobs (Lawler *et al.*, 1982; Grabowski *et al.*, 1986). Shi and Pan (2006) present three improved branch-and-bound algorithms based on Carlier algorithm and other existing techniques that substantially outperform Carlier algorithm in terms of CPU time and robustness.

Schrage algorithm, a heuristic method suggested by Schrage (1971), is an effective algorithm that is used in single machine problem. The algorithms of McMahon and Florian (1975) and Carlier(1982) are based on Schrage algorithm (Lenstra (1977), Carlier (1982) and Potts (1980) for generating a good solution of the $1|r_j, q_j|C_{\max}$ problem). 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 single machine problem. This can lead to decrease computational efforts in the job shop problems (Wenqi and Aihua, 2004).

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 (1982) shows a sequence, in this problem, with a conjunctive graph G = (X, U). The set X of nodes is obtained by adding two nodes O and * to the set i of jobs: $X = I \cup \{O,*\}$, where, O is a job beginning and * a job end. The set U of arcs includes three sets: $U = U_1 \cup U_2 \cup U_3$. 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.

Schrage algorithm (Carlier, 1982): 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 $\bar{U}$ is the set of jobs to be scheduled and t is the time. The steps of this algorithm are shown in Fig. 3.

In Schrage algorithm, presented by Carlier(1982), 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 (Wenqi and Aihua, 2004; Carlier, 1982). According to the Step 2 of the Schrage algorithm, in each

```
Step 1: $t = \underset{i \in I_i}{\mathrm{Min}}(r)$; $U = \Phi$
Step 2: At time t, select a job from $\bar{U}$ with $r_i \le t$, such as job j with
        the largest $q_i$ (if there is a tie, breaks it with the largest $p_i$ and
        if there is still a tie, breaks it with random selection), then set
        $t_j = t$ and $U = U \cup \{j\}$.
Step 3: Set $t = \max\{t_j + p_j; \underset{i \in I\bar{U}}{\mathrm{Min}}(r)\}$.
        If U = I, stop
        Otherwise, return to Step 1.
```

Fig. 3: Schrage algorithm

```
Step 1: $t = \min\{r_i; i \in V_k\}$ and R = Vk.
Step 2: If $r_i > t$, then $u_i = q_i\ \delta \times (r_i - t)$; otherwise, $u_i = q_i$, where $i \in R$.
Step 3: Choose a job from R, say j and with the greatest $u_j$ (If there
        are ties, break them by giving preference to the greatest $q_j$. If
        there are still ties, break them by giving preference to the
        greatest $p_j$). Set $t_j = \max\{t; r_j\}$, $R \Leftarrow R \setminus \{j\}$.
Step 4: If $R = \Phi$, stop; otherwise, set $t = \max\{t_j + p_j; \min\{r_i; i \in R\}\}$
        and return Step 1.
```

Fig. 4: Schrage algorithm with DS

stage, only the ready jobs, ($r_i \le t$) from set $\bar{U}$, are sequenced. In the following we denote these jobs by the set R. However, it is possible that there exist a job k ($k \in \bar{U}$) 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 $\bar{U}$ so that the jobs with large tail that are not ready have chance to be selected.

Schrage algorithm with DS (Wenqi and Aihua, 2004): For solving a single machine problem the Schrage algorithm is easy to be implemented, but the quality of the solution that it produces usually leaves much room for improvement. Wenqi and Aihua (2004) improved the Schrage algorithm and presented a more effective algorithm with a disturbance (DS). They stated that in the Schrage algorithm when the priority of r_i, p_i, q_i not respected strictly, much better solutions are probably obtained. In the situation that the value of r from a job is small while the value of q from this job is not large enough, the algorithm might delay this job. Therefore, they applied a disturbance (DS) on the priority rules based on Schrage algorithm. The steps of Schrage algorithm with DS are shown in Fig. 4.

Here, δ is disturbance coefficient. The degree of disturbance that determines the performance of DS is affected by the value of the coefficient δ. So, the suitable selection of δ in DS is an important issue. The experiments show that if the value of δ is too small or too large, the performance of DS is poor (Wenqi and Aihua, 2004). The former makes the machines idle too frequently to get enough rewards and the latter makes DS degenerate into the Schrage algorithm. DS probably obtains the best solutions when the value of δ is between 1.0 and 6.0 (in their experiments, it is seldom over 6.0) and it tends to obtain the same solution with different values of δ.

The semantic of the Schrage algorithm with disturbance presented is correct, i.e., in the situation that the value of r of a job is small while the value of q of this job is not large enough, the algorithm might delay this job. In this algorithm, in each stage, all of the jobs can be sequenced (i.e. $R = \bar{U}$) and the tail of the jobs that are not ready (their ready time (r_i) is larger than t), is adjusted by $\delta \times (r_i - t)$. The decision criterion for sequencing is u_i, where, $u_i = q_i - \delta \times (r_i - t)$ for $r_i > t$ and $u_i = q_i$, for $r_i < t$. But there are three main problems in modeling of DS. First, decision criterion u_i can not be sufficient alone for sequencing. For example if we have job 1 with $r_1 = 10$, $p_1 = 4$, $q_1 = 10$ and job 2 with $r_2 = 15$, $p_2 = 10$, $q_2 = 41$, by this algorithm; $u_1 = 10$, $u_2 = 11$ (assumed that δ to be in extreme i.e., 6), then job 2 is sequenced earlier than job1. This result is wrong and can be lead to increase the makespan. Because when $r_2 > r_1 + p_1$, there is not any rational reason to delay job 1. Therefore, we need a more comprehensive decision criterion to resolve this important issue. Second, in each stage, all of the unscheduled jobs are checked; this problem increases the computational efforts in the large scale problems. Third, the quality of solution depends on coefficient δ but there is not any semantic for selecting this coefficient. Also, if the coefficient δ varies, then the computational effort will be high.

Modified Schrage algorithm: As stated in the previous section, an important area for improvement in the Schrage algorithm is considering the jobs with large tail even if they are not ready. It means that we must expand the set R by adding those jobs to the set. But the main problem rises which jobs can be added? In other words, in Schrage algorithm, if $r_i < r_j$ and $q_i > q_j$, i is earlier sequenced j that is logically true and proved in literature (Carlier, 1982). But if $r_i < r_j$ and $q_i < q_j$ 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 $0 \le r_i < r_j$ and $q_i < q_j$, a necessary condition for sequencing j before i is ($q_j > m + q_i$ and $p_i > m$) where, $r_j - t = m$ and $p_i \ge 0$.

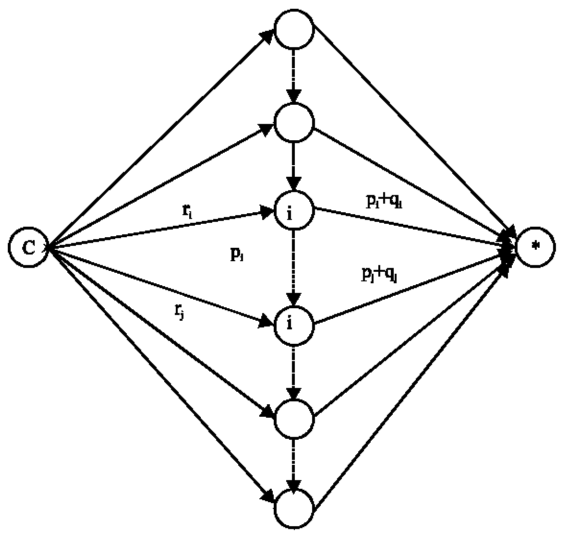

Fig. 5: Associated conjunctive graph

Proof: We demonstrate the sequence of jobs with a conjunctive graph G = (X, U) similar to Carlier algorithm (Fig. 5).

Let U be the set of jobs already scheduled, Ū be the set of all other jobs and c_k 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 Ū. Let t be max (c_k, r_i). Assume that the Job j is a job with r_j so that $r_j > t$ and $q_j > q_i$. We prove that a necessary condition for sequencing j before i is: $q_j > m+q_i$ and $p_i > m$, where, r_j-t = m and $p_i >= 0$. Here, there exist two options as follows:

Option 1: job i to be scheduled before job j
Let L_i^1 be the length of path that pass through 0, 1, i, * and L_j^1 be the length of path that pass through 0, 1, j, * and $L_1 = \text{Max}(L_i^1, L_j^1)$. Thus we have:

$$L_1 = \text{Max}(t+p_i+q_i,\ t+p_i+p_j+q_j) \quad (1)$$

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 $L_2 = \text{Max}(L_i^2, L_j^2)$. Then,

$$L_2 = \text{Max}(r_j+p_j+q_j,\ r_j+p_j+p_i+q_i) \quad (2)$$

Assume that job j is sequenced earlier than job i thus L_2 must be lower than L_1 because the aim is to minimize the longest path through 0 to *. Therefore, the formula (3) should be true.

$$\text{Max}(r_j+p_j+q_j,\ r_j+p_j+p_i+q_i) < \text{Max}(t+p_i+q_i,\ t+p_i+p_j+q_j) \quad (3)$$

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

$$\text{Max}(r_j+p_j+q_j,\ r_j+p_j+p_i+q_i) < t+p_i+q_i \quad (4)$$

$$\text{Max}(r_j+p_j+q_j,\ r_j+p_j+p_i+q_i) < t+p_i+p_j+q_j \quad (5)$$

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

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

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

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

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

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

Let $$r_j-t = m \Rightarrow r_j = m+t \quad (10)$$

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

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

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

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

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

By the assumption: $p_j \geq 0$ and $m \geq 0$, thus, the formula (12) is false. That means the formula (7) is false, then the formula (4) is false all the times. The formulae (10) and (11) by the assumption are true. That means the formulae (8) and (9) are true. So the formula (5) is also true all the times. Thus, the formula (3) is true that means $L_2 < L_1$ 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 ($q_j > m + q_i$ and $p_i > m$). But if the jobs j is not critical then sequencing it before i can be lead to increase delay (r_j-t) and makespan. We define historically a condition for criticality of the job j by following.

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

Proposition 1: In stage s and k ∈ Ū

```
Step 1: Let $t = \underset{i \in I_1}{\text{Min}}\, r_i$, $p = p_l$; $U = \Phi$, $l$ is index of $t = \underset{i \in I_1}{\text{Min}}\, r_i$
Step 2: Find $r_i <= t+p$; $i \in \bar{U}$, If $r_i \le t$, then $u_i = q_i$, elseif $r_i > t$ and
        $t+p+p_i+q_i > t + \sum_{k \in \bar{U}_i} p_k + \underset{k \in \bar{U}_i}{\text{Min}}\, q_k$, $k \in \bar{U}$ then $u_i = q_i - (r_i - t)$;
        otherwise $u_i = 0$.
Step 3: 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
        $t_j = \max\{t; r_j\}$, $U = U \cup \{j\}$.
Step 4: Set $t = \max\{t_j + p_j; \underset{i \in \bar{U}}{\text{Min}}\, r_i\}$. If $U = I$, stop; otherwise, return
        to Step 2.
```

Fig. 6: Modified Schrage algorithm

$$h(s) = t_s + \sum_{k \in \bar{U}_i} p_k + \underset{k \in \bar{U}_i}{\text{Min}}\, q_k$$

is a lower bound on the optimal makespan.

Proof: Carlier (1982) proved that for all $I_1 \subseteq I$,

$$h(I_1) = \underset{i \in I_1}{\text{Min}}\, r_i + \sum_{i \in I_1} p_i + \underset{i \in I_1}{\text{Min}}\, q_i$$

is a lower bound on the optimal makespan. Since $\bar{U} \subseteq I$, therefore h(s) is a lower bound on the optimal makespan. Now we present a modified Schrage algorithm according to theorem 1. the steps of algorithm are shown in Fig. 6.

In this algorithm, in each stage, both ready jobs and the jobs with large tails (based on theorem 1) 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.

We coded the three algorithms: Schrage Algorithm (SA), Schrage algorithm with disturbance (DS) and Modified Schrage Algorithm (MSA), with MATLAB and have tested 1000 problems. For each problem with n jobs, 3n integers have been generated with uniform distributions between 1 and r_{max}, p_{max}, q_{max} (Carlier, 1982).

We set $p_{max} = 50$, $r_{max} = q_{max} = n.k$ and examined 50 values for k and 20 values for n = 50, 100 …., 1000. The problems are solved by the above three algorithms and enumerated the best solutions for problems (minimum makespan). Table 1 shows the comparison of the best solution between our algorithm and two others and the CPU time for running 1000 problems for each algorithm.

Of the 1000 problems, MSA gets the best solution in 880 cases. The results show that MSA got better results than SA and DS. The CPU time in SA is less than the other algorithms and DS spent too much CPU time. Since DS algorithm in each stage, computes the measure u for the remaining jobs unscheduled and checks all of them, thus in large scale problems it makes more

Table 1: Results of three algorithms on 1000 random problems

	SA	DS	MSA
No. of the best solution	450	690	880
Average CPU time for running 1000 problem (second)	56.9	563	151

computing efforts. In the proposed algorithm using the proven theorem, computing the measure u is limited to a few jobs and lead to decreasing CPU time and the complexity. M.A.S gets much better solutions than DS and SA and can be ignored a few increasing in CPU time respect to SA.

Proposed Approach for job shop scheduling problem: The SB algorithm sequences the bottleneck machine first while ignoring the remaining unscheduled machines. After the machine is scheduled, the reoptimization procedure is triggered. The SB repeats the single machine scheduling procedure until all machines have been scheduled. subproblem solution procedure and reoptimization are two important factor in SB approach. In the previous section, we presented an efficient heuristic for subproblem solution that can decrease computational efforts. This study 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 to 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 would have delayed.

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

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

Subproblem solution procedure is base on modified Schrage algorithm presented in the pervious section. In this procedure, the prioritization is base on the head and tail of the operations (u_{ij}), however as sated above, we should prioritize the operations of critical machines. 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:

```
Step 1: Let $t = \min_{o_{ij}\in I_j} r_{ij}$
Step 2: $p = p_l$; $l$ is index of $\min_{o_{ij}\in I_j} r_{ij}$
        Find $r_{ij} \le t+p$; $o_{ij}\in I_j$,
        If $r_{ij} \le t$, $o_{ij}\in V_j$, then $u_{ij} = q_{ij}$,
        Elseif $r_{ij} \le t$, $o_{ij}\notin V_j$, then $u_{ij} = q_{ij}*$ sqrt$(1\text{-}md_j^2)$,
        Elseif $p+p_{ij}+q_{ij} > \sum_{o_{ij}\in I_j} p_{ij} + \min_{o_{ij}\in I_j} q_{ij}$ and $o_{ij}\in V_j$, then $u_i = q_i - (r_i - t)$,
        Elseif $p+p_{ij}+q_{ij} > \sum_{o_{ij}\in I_j} p_{ij} + \min_{o_{ij}\in I_j} q_{ij}$ and $o_{ij}\notin V_j$, then
        $u_{ij} = (q_{ij}-(r_{ij}-t))*$sqrt$(1\text{-}md_j^2)$;
        otherwise $u_{ij} = 0$.
Step 3: 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 $st_{ij} = \max\{t; r_{ij}\}$, $I_j = I_j \cup \{o_{ij}\}$, $U = U \cup o_{ij}\}$, $o_{ij}*$
        is labeled in set $V_j$ otherwise, stop.
Step 4: Set $t = \max\{st_{ij} + p_j; \min_{o_{ij}\in I_j} r_{ij}\}$. If all operations of set $V_j$ are
        labeled , stop;
        otherwise, return to Step 2.
```

Fig. 7: Subproblem solution procedure

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

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 in to account in subproblem solution procedure (Fig. 7). The symbols used in this algorithm are as follows:

N	= No. of jobs
M	= No. of machines
j_i	= Job i
m_j	= Machine j
o_{ij}	= Operation related to job i on machine j
U	= Set of scheduled operations
$\bar{U}$	= Set of unscheduled operations
r_{ij}	= The release time (head) of operation o_{ij}
q_{ij}	= The delivery time (tail) of operation o_{ij}
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 the operation o_{ij}

The proposed algorithm for the job shop problem base on shifting bottleneck is shown in Fig. 8.

Calculations of heads and tails are based on all conjunctive arcs and the fixed disjunctive arcs (Brucker *et al*., 1994).

For the sake of simplicity, we can define tail of operations as follows:

```
Step 1: Let $U = \Phi$
Step 2: Calculate $r_{ij}$, $q_{ij}$.
Step 3: Find critical machine ($m^*$).
Step 4: Find the ready operations of set $\bar{U}$ ($r_{ij} = 0$), cluster the ready
        operations in same machine and find $V_j$.
Step 5: If $V_j \ne \Phi$ sequence the operations of set $V_j$ by the subproblem
        solution procedure, $\forall j \in M$.
Step 6: If $\bar{U} = \Phi$, stop; otherwise, return to step 2.
```

Fig. 8: Proposed approach

Table 2: Comparison of the proposed approach to SB for instance problems

				Shifting bottleneck		Proposed approach	
Problem	n	m	Optimum (LB UB)	Makespan	CPU(S)	Makespan	CPU(S)
FT6	6	6	55	55	0.6	55	0.6
FT10	10	10	930	1015	4.0	1024	2.8
FT20	20	5	1165	1290	1.4	1346	4.0
ABZ5	10	10	1234	1306	2.3	1324	1.9
ABZ6	10	10	943	962	5.1	978	4.1
ABZ7	20	15	656	730	47.5	742	26.8
ABZ8	20	15	(645 669)	774	50.5	780	28.4
ABZ9	20	15	(661 679)	751	37.7	778	25.7

$$q_{ij} = \sum_{o_{ij}\in S_{ij}} p_{ij} \qquad (18)$$

This approach generates a feasible solution for job shop problem. Then, we can use reoptimization phase of SB approach for improving solutions.

EXPERIMENTAL RESULTS

The algorithm is coded in MATLAB. We used a set of job shop scheduling problems from benchmark problems (Fisher and Thompson, 1963; Adams *et al*., 1988; Wenqi and Aihua, 2004). The proposed approach is compared by SB procedure. Comparisons are base on two main factors; solution quality (makespan), CPU time as computations efforts. The results of experiments are represented in Table 2.

Experimental results show that our method in small size until 6×6 gives us optimal solution. Also in large scale, the algorithm gives us good solutions. CPU time in our method is much lower than SB procedure while the makesapn of the proposed approach is comparable with SB. In sum, Experiment results show the superiority of our approach in comparison to SB in large scale problems especially in computational efforts. This approach can be provided a good seed in the iterative methods as a initial solution. Presenting hybrid approaches by iterative heuristics such as tabu search, simulated annealing and genetic algorithm can be future researches.

CONCLUSION

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 study, first, presents a modified Schrage algorithm for single machine scheduling problems with heads and tails that is an effective subproblem solution procedure. Then, it presents a heuristic approach for job shop scheduling that resolve reoptimization difficulties in SB. Finally, the proposed algorithm is tested and validated. Experiment results show the superiority of our approach in comparison to SB in large scale problems especially in computational efforts. This approach can be a good initial seed in using search techniques. Presenting hybrid approaches by iterative heuristics such as tabu search, simulated annealing and genetic algorithm can be future researches.

REFERENCES

Adams, J., E. Balas and D. Zawack, 1988. The shifting bottleneck procedure for job shop scheduling. Manage. Sci., 34 (3): 391-401.

Applegate, D. and W. Cook, 1991. A computational study of job shop scheduling problem. ORSA. J. Comput., 3 (2): 149-156.

Baker, K.R. and Z.S. Su, 1974. Sequencing with due dates and early start times to minimize tardiness. Naval Res. Logistics Q., 21: 171-176.

Balas, E., 1969. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. Operat. Res., 7 (6): 941-957.

Balas, E., J.K. Lenstra and Vazacopoulos, 1995. The one machine problem with delayed precedence constraints and its use in job shop scheduling. Manage. Sci., 41 (1): 94-109.

Brinkkotter, W. and P. Brucker, 2001. Solving open benchmark instances for the job shop problem by parallel head-tail adjustments. J. Scheduling, 4 (1): 53-64.

Brucker, P., B. Jurisch and B. Sievers, 1994. A branch and bound algorithm for the job shop scheduling problem. Discrete Applied Math., 49: 107-127.

Carlier, J., 1982. The one-machine sequencing problem. Eur. J. Operat. Res., 11: 42-47.

Carlier, J. and E. Pinson, 1994. Adjustments of heads and tails for the job-shop problem. Eur. J. Operat. Res., 78: 146-161.

Dauzere-Peres, S. and J.B. Lasserre, 1993. A modified shifting bottleneck procedure for job-shop scheduling. Int. J. Prod. Res., 31 (4): 923-932.

Demirkol, E., S.V. Mehta and R. Uzsoy, 1997. A computational study of shifting bottleneck procedures for shop scheduling. J. Heuristics, 3: 111-137.

Fisher, H. and D.L. Thompson, 1963. Probabilistic Learning Combinations of Local Job Shop Scheduling Rules. In: Industrial Scheduling, Muth, J.F. and G.L. Thompson (Eds.). Prentice-Hall, Englewood Cliffs, NJ.

Grabowski, J., E. Nowicki and S. Zdrzalka, 1986. A block approach for single-machine scheduling with release dates and due dates. Eur. J. Operat. Res., 26 (2): 278-285.

Holtsclaw, H.H. and R. Uzsoy, 1996. Machine criticality measures and subproblem solution procedures in shifting bottleneck methods: A computational study. J. Operat. Res. Soc., 47: 666-677.

Lawler, E.L., J.K. Lenstra and A.H.G. Rinnooy Kan, 1982. Recent Developments in Deterministic Sequencing and Scheduling: A Survey. In: Deterministic and Stochastic Scheduling, Dempster, M.A.H., J.K. Lenstra and A.H.G. Rinnooy Kan (Eds.). Reidel, Dordrecht, 1982.

Lenstra, J.K., 1977. Sequencing by Enumerative Methods. Mathematical Centre Tracts 69, Mathematisch Centrum, Amsterdam.

McMahon, G.B. and M. Florian, 1975. On scheduling with ready times and due dates to minimize maximum lateness. Operat. Res., 23: 415-482.

Mukherjee, S. and A.K. Chatterjee, 2006. On the representation of the one machine sequencing problem. Eur. J. Operat. Res, doi:10.1016/j.ejor.2006.07.024.

Potts, C.N., 1980. Analysis of a heuristic for one machine with release dates and delivery times. Operat. Res., 28: 1436-1441.

Schrage, L., 1971. Obtaining optimal solutions to resource constrained network scheduling problem. Unpublished Manuscript.

Shi, L. and Y. Pan, 2006. Branch-and-bound algorithms for solving hard instances of the one machine sequencing problem. Eur. J. Operat. Res., 168: 1030-1039.

Uzsoy, R. and C.S. Wang, 2000. Performance of decomposition procedures for job shop scheduling problems with bottleneck machines. Int. J. Prod. Res., 38: 1271-1286.

Vakhania, N., 2004. Single-machine scheduling with release times and tails. Ann. Operat. Res., 129: 253-271.

Wenqi, H. and Y. Aihua, 2004. An improved shifting bottleneck procedure for the job shop scheduling problem. Comput. Operat. Res., 31: 2093-2110.

Wu, C.S., D.C. Li and T.I. Tsai, 2006. Applying the fuzzy ranking method to the shifting bottleneck procedure to solve scheduling problems of uncertainty. Int. J. Adv. Manuf. Technol., 31: 98-106.