Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Job Shop Scheduling with Earliness and Tardiness Penalties Using Iterative Repair Method

Te-Wei Chiang and Hai-Yen Hau
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan 10617, R.O.C.

Abstract

In this paper, we propose an approach based on iterative repair for job-shop scheduling problems. This approach had been successfully applied to railway scheduling problems. To show the flexibility of this approach, it has been adapted to solve to the job-shop scheduling problems with weighted early/tardy objective in this paper. The weighted early/tardy objective is compatible with the philosophy of just-in-time (JIT) production to minimize inventory costs. Since the objective function is not a regular performance measure (i.e., a function that is non-decreasing in the completion times), it can not be solved by the conventional theorems on job ordering. The repair-based approach is successfully applied to solve the problems. Experimental results show the efficiency and effectiveness of this approach.

1. INTRODUCTION

1.1 The Job Shop Scheduling Problem

The elements of a job shop scheduling problem [2] are a set of machines and a collection of jobs to be scheduled. Operation precedence constraints give the order in which the operations that comprise each job must be processed. The job shop scheduling problem thus can be defined as the allocation of machines over time to perform a collection of jobs to maximize(minimize) a performance criterion while satisfying the operation precedence constraints and resource constraints.

The objectives of scheduling are often multidimensional, and there are many possible measures of scheduling performance. Generally speaking, however, two factors are of primary interest: shop time and due-date performance. In actual shops, meeting due-dates tends to be a more important criterion than minimal shop time. Unfortunately, the study of due-date performance is also much more complicated. There is no single, universally-accepted measure of effectiveness on this dimension. The job shop literature suggests several measures such as weighted tardiness and weighted earliness and tardiness. For the most part, the literature of scheduling has been confined to problems involving penalty functions which are non-decreasing in job completion times. They refer to such functions as *regular performance measures* [11].. In spite of the importance of nonregular performance measures, such as weighted early/tardy objective, very little analytical work has been done in this area. Many important theorems on job ordering, cannot be applied to these problems in order to make optimization techniques such as branch-and-bound and dynamic programming more efficient.

In this paper, we examine the job shop scheduling problem to minimize total early and tardy costs. More specifically, the objective is to find a schedule that minimizes the total earliness and total tardiness costs of all jobs, subject to the constraints that no preemption of jobs is allowed and all jobs are initially available. This problem is a generalization of the weighted tardiness problem. The early/tardy objective is compatible with the philosophy of just-in-time (JIT) production to minimize inventory costs by producing goods as close to their due dates as possible.

1.2 Repair-Based Approach

There is a long history of AI programs that use repair or debugging strategies to solve problems. Minton et al. [16] suggested a repair method based on min-conflicts heuristic for constraint satisfaction and scheduling problem problems. Zweben et al. [19] proposed a general scheduling system being applied to the Space Shuttle ground processing problem based on iterative repair. Chiang and Hau proposed an iterative repair approach for railway scheduling problems [7]-[9] and job-shop scheduling problems [10]. In repair-based approach, one starts with a complete but possibly infeasible schedule and searches through the space of possible repairs. The search can be guided by a repair heuristic, such as min-conflicts heuristic [19] that attempts to

minimize the number of constraint violations after each step. The heuristic can be used with a variety of different search strategies such as *Local Search* [14], [18], *Simulated Annealing* [1], and *Tabu Search* [6], [12], [13].

1.3 Overview of the Paper

In this paper, we propose a job-shop scheduling algorithm based on iterative repair method. There are four basic components in the proposed system: Initial Scheduler, Repair Scheduler, Local Scheduler and Conflict Management. First, the Initial Scheduler heuristically generates an initial schedule without considering constraint violations. The Repair Scheduler then uses an *Earliest-Conflict-First* heuristic to repair conflicts sequentially according to the time it occurs in the timetable after the Conflict Management finds all conflicts in the initial schedule. In local scheduling, the Local Scheduler resolves the conflict given by Repair Scheduler using one of the proposed three repair methods while minimizing the objective function. Thus the repair scheduler and the local scheduler cooperate in harmony to generate the complete schedule. During each repairing iteration, local search techniques aid the selection of an appropriate repair method such that the objective function is minimized.

2. SURVEY

2.1 Problem Description

For convenience, the notations to be used in this paper is shown as follows, where operation j of job i is referred to as operation (i, j).

N : Number of jobs.
M : Number of machines.
r_i : Ready time of job i.
s_{ij} : Start time of operation (i, j).
c_{ij} : Completion time of operation (i, j).
p_{ij} : Processing time of operation (i, j).
m_{ij} : Machine selected to process operation (i, j).
d_i : Due date of job i.
d_{ij} : Due date of operation (i, j).
T_i : Tardiness of job i, which is defined as the amount of job completion time c_{iN} passes the job due date d_i, i.e., $\max\{0, c_{iN} - d_i\}$.
E_i : Earliness of job i, which is defined as the amount of job completion time c_{iN} leads the job due date d_i, i.e., $\max\{0, d_i - c_{iN}\}$.

A static and deterministic scheduling problem can now be formulated as follows:

P : min J,

subject to

1) precedence constraints:

$$c_{ij} \le s_{i,j+1} \quad (i=1, 2, ..., N;\ j=1, 2, ..., M\text{-}1) \qquad (1)$$

2) capacity constraints:

$$\text{if } m_{ij} = m_{i'j'}, \text{ then } s_{ij} - s_{i'j'} \ge p_{i'j'} \text{ or } s_{i'j'} - s_{ij} \ge p_{ij}$$
$$(i,i'=1, 2, ..., N;\ i \ne i';\ j,j'=1, 2, ..., M) \qquad (2)$$

3) processing time requirements:

$$c_{ij} - s_{ij} = p_{ij} \quad (i=1, 2, ..., N;\ j=1, 2, ..., M) \qquad (3)$$

4) ready time requirements:

$$s_{i1} \ge 0 \quad (i=1, 2, ..., N) \qquad (4)$$

Corresponding to these constraints are four types (Type I - Type IV) of conflicts that may arise during the scheduling process, each of which corresponds to one of the constraint type defined above. For instance, Type I conflict corresponds to the violation of precedence constraint. Each conflict is associated with either one operation (Type III and Type IV) or two operations (Type I - Type II).

The objective we consider here is the minimum weighted earliness and tardiness, i.e.

$$J \equiv \sum_{i=1}^{N} (u_i E_i + v_i T_i) \qquad (5)$$

where the weight u_i times the job earliness T_i represents the job earliness penalty and the weight v_i times the job tardiness T_i represents the job tardiness penalty. This objective function is compatible with the philosophy of just-in-time (JIT) production to minimize inventory costs [5] and is a non-regular performance measure [11].

The following assumptions are made for the problem: (1) all the jobs are available at time zero; (2) operation processing is assumed to be nonpreemptive; (3) processing time of the jobs on the machines are known beforehand.

2.2 Dispatching Rules

Dispatching rules are a distributed sequencing strategy, by which a priority is assigned to each job waiting for service on a machine: whenever a machine is free, the one with highest priority is selected. There is a large number of such rules, oriented to different performance measures [2], [11]. Some dispatching rules are optimal for the single machine case; others are based on heuristic insights into the scheduling problem. Here the following

rules will be considered: their purpose will be to provide a comparison for the proposed scheduler.

1) The *Shortest Processing Time* (SPT) rule, which gives priority to the smallest processing time.
2) The *Largest Processing Time* (LPT) rule [17], which gives priority to the largest processing time.
3) The *Earliest Due Date* (EDD) rule, which gives priority to the earliest due-date. EDD rule can obtain optimal schedules for the single machine problems [11] with the performance measures such as the maximum lateness and maximum tardiness.
4) The *Operation Due Date* (ODD) rule, which gives priority to the earliest operation due-date. There are alternative rules for setting operation due-dates. If we use the subscripts (i, j) to denote the ith operation of job j and adopt the convention $d_{0j} = r_j$, then we have:

$$\text{CON:} \quad d_{ij} = d_{i-1,j} + a_j / M \tag{6}$$
$$\text{TWK:} \quad d_{ij} = d_{i-1,j} + p_{ij} a_j / \Sigma_i p_{ij} \tag{7}$$

where a_j represents the original flow allowance, which is defined as $a_j = d_j - r_j$.

5) The *Modified Operation Due date* (MOD) rule [4], which gives priority to the earliest modified operation due-date. An operation's modified due-date is defined as its original operation due-date or its early finish time, whichever is larger. MOD is currently the best known rule for the unweighted tardiness criterion, except for SLK/OPN at a loosely set due date [3].
6) The *Operation Priority Index* (OPI) rule [15], which gives priority to the smallest operation priority index. The OPI rule combines process time and its operation due date with equal weight to create an index as follows:

$$\text{OPI}_{ij} = p_{ij} + d_{ij} - t \tag{8}$$

where p_{ij}, d_{ij} and t represent the operation process time, operation due-date and the time at which the dispatching decision is made, respectively.

2.3 Local Search Based Algorithms

local search based algorithms are based on the idea of exploring the set of feasible solutions (schedules) by perturbing a given solution and comparing the new solution with the old one. Basically, a local search based algorithm can be described as follows:

Local Search Based Algorithm

Step 1. Generate an initial schedule.

Step 2. Select the a neighbor from the neighborhood of the current schedule.
  2.1 Apply evaluation function.
  2.2 Check admissibility.

Step 3. Update the schedule status if a schedule is found, otherwise go back to step 2.

Step 4. Repeat steps 2 through 3 until done.

Generally, a neighbor is randomly selected from the neighborhood of the current schedule. The admissibility of a neighbor is decided by the search techniques the algorithm used such as *local search* (*local improvement*), *simulated annealing*, and *tabu search*.

1) Local Search (*local improvement*): Local search [14], [18] is one of the few successful techniques for combinatorial optimization problems (finding the minimum/maximum of a given objective function depending on many decision variables). Assume that vector x represents the set of values assigned to the decision variables and a *neighborhood* $N(x)$ is defined for x. Now given a solution point x_k and an objective function $f(x)$ to be minimized (or maximized), a solution point x_{k+1} is searched from the *neighborhood* $N(x_k)$ of x_k such that $f(x_{k+1}) < f(x_k)$ (or $f(x_{k+1}) > f(x_k)$). If such a point exists, then the similar process is repeated for x_{k+1}. Otherwise, x_k is retained as a *local optimum* with respect to $N(x_k)$. As such, a set of solutions is generated and each of them is locally improved within its neighborhood.

2) Tabu Search : Tabu search was introduced by Glover [12], [13]. The underlying idea is to forbid some search directions (moves) at a present iteration in order to avoid cycling, but to be able to escape from a local optimal point. This strategy can make use of any local improvement technique. It consists a *tabu list* which maintains a memory of moves recently taken in order to prevent reversals which would cycle back to the same local optimum. Furthermore, an *aspiration criterion* which allows override of tabu status for moves which lead to a better solution than previously found was employed to increase the possibility of finding global optimum.

3. PROPOSED ARCHITECTURE

3.1 Problem Formulation

Mathematically, the problem can be formulated as a constrained optimization problem (see section 2). We can transform the constrained optimization

problem to unconstrained optimization problem via the incorporation of the constraint violations into the objective function.

Assume that $\underline{X}$ is a solution point (or a schedule). It can be expressed as

$$\underline{X} = \underline{x}_{ij},\ 1 \le i \le N \text{ and } 1 \le j \le M \tag{9}$$

where N is the number of jobs and M is the number of machines. Each $\underline{x}_{ij}$ is associated with operation (i, j) and corresponds to an start time, completion time, machine triplet $[s_{ij}, c_{ij}, m_{ij}]$. The objective function is

$$C(\underline{X}) = J(\underline{X}) + \lambda Q(\underline{X}) \tag{10}$$

where $J(\underline{X})$ represents the original objective function described in Section 2, which is our measure for schedule quality; $Q(\underline{X})$ is the cost due to the conflicts in schedule $\underline{X}$ and λ is the Lagrange multiplier used to relax the constraint violations. We usually refer the objective function $C(\underline{X})$ to the cost function of the problem. For simplicity, we call $C(\underline{X})$, $J(\underline{X})$ and $Q(\underline{X})$ the total cost, the schedule cost and the conflict cost of the schedule respectively. $Q(\underline{X})$ can be defined as $\Sigma_{k=1}^{K(\underline{X})} q_k$, where q_k is an positive integer representing the time interval the kth conflict is violated, assuming there are $K(\underline{X})$ conflicts in the schedule $\underline{X}$.

3.2 System Architecture

Similar to [9], the proposed system consists of four basic components: Initial Scheduler, Repair Scheduler, Local Scheduler and Conflict Management. First, an initial schedule is heuristically established by Initial Scheduler regardless of constraint violations. The Repair Scheduler then determines the sequence of conflicts to be repaired according to *Earliest-Conflict-First* heuristic after the Conflict Management finds all conflicts in the initial schedule. Then the Local Scheduler resolves the conflict given by Repair Scheduler while minimizing the objective function.

4. PROPOSED ALGORITHM

In this section we will first introduce how to generate an initial schedule and how to repair a conflict. Then we propose an algorithm to repair all conflicts in the initial schedule.

4.1 Initial Schedule Generation

Since the repair-based algorithm is a kind of local search algorithm, we must generate an appropriate initial schedule, such that the resulting schedule is acceptable. From search's point of view, the initial schedule can be regarded as the start point of the search. For the purpose of finding near-optimal solution, the start point must contain some global information. Through the combination of local search and an appropriate start point, the system can quickly find a good conflict-free schedule. Therefore, we generate an initial schedule containing the following characteristics:

1) The schedule has the minimum value of objective function.
2) The schedule satisfies all types of constraints except the capacity constraints. In other words, without considering the capacity constraints, the initial schedule is an optimum one.
3) The process time of each operation must be appropriately distributed in order to reduce the number of capacity constraint violations and hence alleviate the load of the repair-based system.

Therefore, we can divided the scheduling process into two parts: the initial scheduling process and the iterative repairing process. In the initial scheduling process, the capacity constraints have been relaxed and the initial schedule is generated optimally; in the repairing process, the system coordinates the operations to find a good conflict-free schedule.

We propose six methods to generate an initial schedule:

1) Forward Dispatch from Ready Time (FDRT): For each job, (1) the ready time of the job is assigned to the start time of the first operation, and then (2) the completion time of the previous scheduled operation is assigned to the start time of its succeeding operation, and (3) repeat step (2) until all of the operations belonging to the job have been scheduled.

2) Backward Dispatch from Due-Date (BDDD): For each job, (1) the due-date of the job is assigned to the completion time of the last operation, and then (2) the start time of the previous scheduled operation is assigned to the completion time of its preceding operation, and (3) repeat step (2) until all of the operations belonging to the job have been scheduled.

3) Dispatch by CON Rule (DCON): Apply CON rule (see Eq. (6)) to set operation due-dates and then assign the completion times of operations to the operation due-dates.

4) Dispatch by TWK Rule (DTWK): Apply TWK rule (see Eq. (7)) to set operation due-dates and then assign the completion times of operations to the operation due-dates.

5) Dispatch by Constant Slack Time Rule (DCST): As the *slack time* of an job i, S_i, is defined as

$S_i = d_i - r_i - p_i$, where d_i, r_i, and p_i represent the due-date, ready time and total processing time of the job, respectively. Then the completion time of the last operation as the due-date of the job, and insert constant idle time (S_i / M) between each pair of successive operations and between the first operation and ready time.

6) *Dispatch by Modified Constant Slack Time Rule* (DMCST): This rule is the same as that of *DCST* except that the constant idle time becomes (S_i / (M+1)), and the constant idle time also insert between the last operation and the due-date of the job.

Method (1)-(2) are developed intuitively. Method (3)-(4) are motivated by the operation due-date rule CON and TWK. Method (5)-(6) are developed to uniformly distribute the operations along the time axis. Method (2)-(5) are suitable for the weighted earliness and tardiness problems since the initial schedules generated by these methods have zero earliness/tardiness cost. This is because the last operation of each job just match the due-date of the job.

4.2 Repair Methods

Recall that each conflict is associated with either one operation (Type III and Type IV) or two operations (Type I and Type II). The system will try to shift the violated operation left or right on the time axis so long as the conflict is released, rather than exploring many possible alternatives. There are three repair methods that can repair a conflict :

1) *Swap(SP):* Swap the start times of the two violated operations (the processing times of the two operations remain unchanged). This repair method only suitable for capacity constraint violations. As we swap the two operations contributing a capacity constraint violation, the violation may be disappeared or alleviated.
2) *Left-Shift(LS):* Left-shift the violated operation on the time axis such that the constraint is satisfied. This repair method can not be applied to solve violations of ready time requirement.
3) *Right-Shift(RS):* Right-shift the violated operation on the time axis such that the violated constraint is satisfied. This repair method can be applied to any types of constraint violations.

When a conflict arises between two operations, one of the operations will be selected to be moved in an attempt to reduce the *cost* function as much as possible. The heuristic used to select the operation to be moved considers the *suitability* of the operation to the repair method applied to the conflict. For example, if repair method *LS* is selected to repair a given conflict, then the system will left-shift the left operation associated with the conflict, i.e., the operation with earlier start time; on the other hand, if repair method *RS* is selected to repair a given conflict, then the system will right-shift the right operation associated with the conflict.

To facilitate the selection of repair method for a conflict, we specify the priority of each repair method, such that the repair method with higher priority will be tried first. The priority of a *LS* is higher than that of a *RS* because moving operation right on time axis will cause the operation to occupy the resource longer and hence affect the performance of the schedule. Although the *RS* repair method has lower priority, it plays an important role in our repair-based scheduling system. Since *RS* would not create conflict left of the conflict-free boundary on the time axis, they can facilitate the expansion of the conflict-free area. Notice that the process time of each operation is unchanged during the repairing process, i.e., the proposed repair methods will not affect the operation process time. Thus, Type III conflict (violation of process time requirement) would not arise during the scheduling process.

4.3 Iterative Repair

Since the initial schedule is not conflict-free, we must repair the conflicts in the initial schedule. During each iteration, we iteratively search a repair that resolves the conflict given by the *Earliest-Conflict-First* heuristic while minimizing the cost function. To select an appropriate repair method for a conflict, local search techniques can be used. During the kth iteration, we iteratively search a repair that resolves the given conflict and minimizes the cost function at the same time. If a repair reduces the cost function, i.e., the new cost value, c_n, is smaller than c_o, then we accept the repair and assign c_n to c_o, else we try next priority repair until all possible repairs to the conflict has been tried. If no repair method can reduce the cost function for a given conflict, the lowest priority repair method (right shift the violated operation) will be selected to repair the conflict. This is somewhat different from the conventional local search which forbids all possible moves that increase the cost function.

The local search algorithm is shown in Fig. 1, in which the following notations are used :

S_C : The set of conflicts corresponding to current local schedule
S_R : The set of repair methods

Step 1. (Initialization)
1.1 Generate the initial schedule.
1.2 Find all conflicts in the initial schedule and put these conflicts to S_C.
1.3 Evaluate c_o.
1.4 *count* := 0.
Step 2. 2.1 If S_C is empty or *count* > *limit* then stop, else select and delete the earliest conflict from S_C.
2.2 Put all possible repair methods to S_R.
Step 3. 3.1 Select and delete the highest priority repair method from S_R.
3.2 Test to repair the selected conflict.
3.3 Evaluate c_n.
Step 4. If $c_n < c_o$ or S_R is empty, then perform the repair and goto step 5, else goto step 3.
Step 5. 5.1 Update S_C.
5.2 $c_o := c_n$.
5.3 Increase *count* by one.
5.4 Goto step 2.

Fig. 1. The iterative repair algorithm based on local search techniques.

c_o : The cost of the original local schedule.
c_n : The cost of the new generated local schedule
count : The number of iterations
limit : A prespecified number used to limit the number of iterations in the iterative repair process.

The local search algorithms has a tendency of getting stuck at a local optimum or a cycle. For example, there is a capacity constraint violation between two operations at some machine, the earlier operation has been shifted earlier to repair the conflict. But the shift results in another new capacity constraint violation. To repair the new conflict, the operation has to be shifted later and hence the original conflict comes back. Such that a cycle occurs between the two repair operations.

4.4 Cycle Prevention

To prevent cycles, we incorporate a *forbidden list* into the standard local search algorithm. The forbidden list maintains a memory of cost values recently happened. When two schedules have the same cost values, we can recognize that cycle may occur since different schedules usually correspond to different cost values. When a repair method results in a cycle, we try next priority repair method unless the repair method is the last repair method. This approach is similar to tabu search (see Section 2.3). Tabu search maintains a tabu list to memorize the moves recently taken in order to prevent reversals which would cycle back to the same local optimum. The main difference is that our approach only rejects the moves that may result in cycles but tabu search rejects all moves that are the same as those recently taken. Moreover, we only memorize the cost values as the elements of our forbidden list; on the other hand, each element of tabu list memories several attributes of a move.

5. EXPERIMENTAL RESULTS

We randomly generated three 10-job 10-machine problems according to the following problem factors:

1) *Ready(Arrival) Time of Each Job:* The ready times of all jobs were set to zero.
2) *Process Time:* The process time of each operation was uniformly distributed between 1 and 10.
3) *Due Dates:* The due date of each job was set to the total processing time of the job times a DSF (due date set factor), the DSF is uniformly distributed between 1 and 3.
4) λ *:* The value of Lagrange multiplier was set to 10.

All experiments were run on a PC 80486-66. Each experiment ran until the resulting schedule was conflict-free. In the following we present the results of these experiments. Table I shows the results for the three problems with weighted earliness and tardiness criterion. For simplicity, the weight of job earliness and the weight of job tardiness in Eq. (5), i.e. u_i and v_i , were set to 1 and 2 respectively. Using the priority rules described in Section 2.2, we can find that the EDD rule generates the best average cost for the three problems. The average cost is 339.7. Table I also shows results of the three problems solving by the repair-based approach with different initial schedule generation methods. The forbidden list size is set to 15. We find that the

TABLE I
Experimental Results for the minimum weighted earliness and tardiness problems (10-job 10-machine).

Problem No.	SPT	LPT	EDD	ODD	MOD	OPI	RBA-1		RBA-2		RBA-3		RBA-4		RBA-5		RBA-6	
	cost	cost	cost	cost	cost	cost	cost	time	cost	time	cost	time	cost	time	cost	time	cost	time
1	418	487	314	360	360	366	374	9.3	300	11.6	101	8.3	100	6.9	82	8.7	84	9.5
2	398	365	308	255	255	360	340	8.6	318	15.1	103	8.6	67	8.5	102	12.7	223	13.2
3	563	425	397	426	431	444	545	18.7	168	5.4	223	12.1	99	14.2	214	11.5	203	13.8
Ave.	459.7	425.7	339.7	347	348.7	390	419.7	12.2	262	10.7	142.3	9.6	88.6	9.8	132.6	10.9	170	12.1

cost : the value of the objective function.
time : CPU time.
RBA-1 : repair-based approach using FDRT.
RBA-2 : repair-based approach using BDDD.
RBA-3 : repair-based approach using DCON.
RBA-4 : repair-based approach using DTWK.
RBA-5 : repair-based approach using DCST.
RBA-6 : repair-based approach using DMCST.

repair-based approach with DTWK generates the best average cost, 88.6, which is much better than that generated by EDD priority rule. The average running time is 9.8 seconds.

The experimental results reveal that the repair-based approach with DTWK initial schedule generation method performs better than all of the priority rules in schedule quality. This is because the priority dispatching rules are too myopic; on the other hand, the initial schedule generation method DTWK provides sufficient global information of the search space and hence yields better quality schedules.

6. CONCLUSION

In this paper, we demonstrated a job-shop scheduling system based on the iterative repair method. We introduced how to transform the job-shop scheduling problem into a repair-based search problem. Through the cooperation of the Earliest-Conflict-First heuristic, local search techniques and the cycle prevention scheme, a given infeasible schedule will be efficiently repaired. The system has been applied to the job-shop scheduling problems with weighted early/tardy objective which can not be solved by the conventional theorems on job ordering. Experimental results revealed the effectiveness of our approach for the problem. Besides, due to the characteristic of the repair-based approach, it can easily be applied to the dynamic rescheduling problems. We can modify the original schedule to react to the real status of shop flow, and feed the modified schedule into the repair-based system. The system will automatically repair the conflicts in the schedule. In conclusion, the proposed repair-based system can resolve the job-shop scheduling problem in an efficient and effective manner.

REFERENCE

[1] E. Aarts and J. Korst, "Simulated annealing and boltzmann machines - a stochastic approach to combinatorial optimization and neural computing," *John Wiley & Sons, New York*, 1989.

[2] K. R. Baker, "Introduction to sequencing and scheduling," *John Wiley & Sons*, 1974.

[3] K. R. Baker and J. J. Kanet, "Job shop scheduling with modified due dates," *J. Oper. Management*, vol. 4, no. 1, pp.11-22, 1983.

[4] K. R. Baker, "Sequencing rules and due-date assignments in a job shop," *Management Science*, vol. 30, no. 9, pp.1093-1104, 1984.

[5] K. R. Baker and G. D. Scudder, "Sequencing with earliness and tardiness penalties: a review," *Operations Research*, vol. 38, no. 1, pp. 22-36, 1990.

[6] P. Brandimarte, "Routing and scheduling in flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, pp.157-183, 1993.

[7] T. W. Chiang and H. Y. Hau, "Railway scheduling system using repair-based approach," in *Proc. IEEE Int. Conf. on Tools with Artificial Intelligence*, Washington DC, pp. 71-78, 1995.

[8] T. W. Chiang and H. Y. Hau, "Cycle detection in repair-based railway scheduling system, " in *Proc. IEEE Int. Conf. on Robotics and Automation*, Minnesoda, vol. 3, pp. 2517-2522, 1996.

[9] T. W. Chiang and H. Y. Hau, "Repair-based railway scheduling system with cycle detection, " *IEICE Trans. on Inf. & Syst.*, vol. E79-D, no. 7, July, 1996.

[10] T. W. Chiang and H. Y. Hau, "Job shop scheduling system using repair-based approach," in *Proc. IEEE Int.Conf. on Systems, Man, and Cybernetics*, Bejing, vol. 2, pp. 1525-1530, 1996.

[11] S. French, "Sequencing and scheduling: an introduction to the mathematics of the job-shop," *John Wiley & Sons*, 1982.

[12] F. Glover, "Tabu search - part I," *ORSA J.*

Computing, vol. 1, pp. 190-206, 1989.

[13] F. Glover, "Tabu search - part II," *ORSA J. Computing*, vol. 2, pp. 4-32, 1990.

[14] J. Gu, "Local search for satisfiability (SAT) problem," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, no. 4, pp.1108-1129, 1993.

[15] M. H. Han and L. F. Mcginnis, "Due dates in a manufacturing shop: an unweighted case," *Annals of Operations Research*, vol. 17, pp.217-232, 1989.

[16] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems," *Artificial Intelligence*, vol. 58, pp. 161-205, 1992.

[17] R. Rachamadugu, "Scheduling jobs with proportionate early/tardy penalties," *IIE Trans.*, vol. 27, pp. 679-682, 1995.

[18] R. Sosic and J. Gu, "Efficient local search with conflict minimization: A case study of the n-queens problem," *IEEE Trans. on Knowledge and Data Engineering*, vol. 6, no. 5, pp. 661-668, 1994.

[19] M. Zweben, E. Davis, B. Daun, and M. J. Deale, "Scheduling and rescheduling with iterative repair," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, no. 6, pp. 1588-1596, 1993.