can be computed. It is found that the cumulative variance out of four dominant PCs accounts for more than 99% of total variance.

A solid reference database must be provided for satisfactory Raman studies. With enough data, multivariate analysis can be demonstrated via clustering. Raman spectra with similar PC scores are close to each other. PC scores from normalized intrinsic Raman spectra can be sketched in scatter plots. With limited datasets, Fig. 10 shows only the preliminary PCA results, where fresh samples from mice livers, lungs, kidneys, and glands are differentiated. Meanings of symbol and color are shown in Table V. Scatter plots of up to four PCs are demonstrated. The dominant PC scores are plotted, such as PC1 versus PC2, PC1 versus PC3, PC3 versus PC4, etc. Results from scatter plots testify to the effectiveness of the proposed approach on sample characterization.

VIII. Conclusion

A systematic intelligent-control approach is implemented for biomedical sample characterization. Fuzzy filtering is used for noise filtering. Spectroscopic functions are selected to identify background fluorescence function, and its baseline spectrum is optimized using genetic algorithm. Via the SNV method, each set of data can be normalized in a similar way. The PCA approach is employed for computation of dominant PCs. Data clustering is subsequently applied for further sample differentiation. An example for fresh sample classification is provided offline among normal mice livers, lungs, kidneys, and glands. This approach has no technical difficulty in real-time medical diagnosis, and it can also be extended to the decision making of normal, benign, and abnormal samples. A thorough systematic intelligent-control method for Raman spectroscopic sample characterization has been formulated. Future Raman study direction will be focused on real-time applications for robotic surgery.

References

[1] M. Sonerira, R. Pueyo, and S. Moreno, "Raman spectra enhancement with a fuzzy logic approach," *J. Raman Spectroscopy*, pp. 599–603, 2002.

[2] Z. Ye, "Research on automotive idle speed control and fuzzy control application," M.S. thesis, Tsinghua Univ., Beijing, China, 1996.

[3] Z. Ye and G. Auner, "Linear filtering and nonlinear fuzzy logic filtering for sample identification with Raman spectroscopy," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Washington, DC, Oct. 2003, Available: CDRom.

[4] ——, "Raman spectrum baseline identification and optimization using genetic algorithm," in *Proc. IEEE Int. Symp. Intell. Control*, Taipei, Taiwan, R.O.C., Sep. 2004, pp. 489–494.

[5] R. Wolthuis and T. Bakker, "Raman spectroscopic methods for *in vitro* and *in vivo* tissue characterization," *Fluores. Lumines. Probes BIO Activity*, pp. 433–455, 1999.

[6] M. Hassoun, *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.

[7] R. Barnes *et al.*, "Standard normal variate transformation and detrending of near-infrared diffuse reflectance spectra," *Appl. Spectroscopy*, vol. 43, no. 5, pp. 772–777, 1989.

[8] M. Bacci, S. Porcinai, and B. Radicati, "Principal component analysis of near-infrared spectra of alteration products in calcareous samples," *Appl. Spectroscopy*, vol. 51, no. 5, pp. 700–706, 1997.

[9] Z. Ye and G. Auner. Principal component analysis (PCA) for biomedical sample identification. presented at *Proc. IEEE Int. Conf. Syst., Man, Cybern.*. [CD-ROM]

[10] F. Vogt and M. Tacke, "Fast principal component analysis of large data sets based on information extraction," *J. Chemometrics*, vol. 16, pp. 562–575, 2002.

[11] D. Rusak and L. Brown, "Classification of vegetable oils by principal component analysis of FTIR spectra," *J. Chem. Educ.*, vol. 80, no. 5, pp. 541–543, 2003.

[12] R. Henrion, "N-way principal component analysis theory, algorithm, and applications," *Chemometrics, Intell. Lab. Syst.*, vol. 25, pp. 1–23, 1994.

[13] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[14] M. Jamshidi and L. Zadeh, *Applications of Fuzzy Logic toward High Machine Intelligence Quotient Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[15] K. Passino and S. Yurkovich, *Fuzzy Control*. Menlo Park, CA: Addison-Wesley Longman, 1999.

[16] J. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

An Efficient Search Method for Job-Shop Scheduling Problems

Leyuan Shi and Yunpeng Pan

***Abstract*—In this paper, we present an efficient search method for job-shop scheduling problems. Our technique is based on an innovative way of relaxing and subsequently reimposing the capacity constraints on some critical operations. We integrate this technique into a fast tabu search algorithm. Our computational results on benchmark problems show that this approach is very effective. Upper bounds for 11 well-known test problems are thus improved.**

Note to Practitioners—Through the work presented in this paper, we hope to move a step closer to the ultimate vision of an automated system for generating optimal or near-optimal production schedules. The peripheral conditions for such a system are ripe with the increasingly widespread adoption of enterprise information systems and plant floor tracking systems based on bar code or wireless technologies. One of the remaining obstacles, however, is the fact that scheduling problems arising from many production environments, including job-shops, are extremely difficult to solve. Motivated by recent success of local search methods in solving the job-shop scheduling problem, we propose a new diversification technique based on relaxing and subsequently reimposing the capacity constraints on some critical operations. We integrate this technique into a fast tabu search algorithm and are able to demonstrate its effectiveness through extensive computational experiments. In future research, we will consider other diversification techniques that are not restricted to critical operations

***Index Terms*—Local search, scheduling, tabu search (TS).**

I. Introduction

Optimal production schedules can increase plant production capacity by maximizing utilization of critical pieces of processing equipment, shifting tasks to less busy equipment when possible, and opportunistically fitting production tasks into otherwise unused time windows. With advances in information and communication technologies and the rapid evolution of E-Commerce, these scheduling problems become

Manuscript received February 27, 2003; revised November 19, 2003. This paper was recommended for publication by Associate Editor C. Chu and Editor N. Viswanadham upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant DMI-0100220, in part by the Air Force Office of Scientific Research under Grant F49620-01-1-0222, in part by Rockwell Automation, and in part by John Deere Horicon Works.

The authors are with the Department of Industrial Engineering, University of Wisconsin-Madison, Madison, WI 53706 USA (e-mail: leyuan@engr.wisc.edu; pany@cae.wisc.edu).

Digital Object Identifier 10.1109/TASE.2004.829418

more and more complicated in terms of problem size and uncertainty. This demands the development of new scheduling algorithms and tools that would enable a firm to produce quickly and flexibly according to customer needs and changes in market conditions.

The job-shop scheduling problem (JSP) has historically been and still is a computationally challenging problem. Due to the intrinsic difficulty of the problem, only approximation algorithms seem to offer some hope from a practical standpoint. Approximation algorithms aim at high-quality solutions, which are not necessarily optimal; this allows these algorithms to run much faster than exact algorithms, which seek the optimum. Approximation algorithms range from heuristics based on *priority dispatching* [1], [2], *shifting bottleneck procedures* [3], [4], to *local search* methods (see the survey by Vaessens *et al.* [5]). In recent years, local search methods have achieved remarkable success in terms of both computation time and solution quality. Among them are *simulated annealing* [6]–[8], *tabu search* (TS) [9]–[12], *genetic algorithms* [13]–[15], and the *guided local search* for JSP [16].

In this short paper, we develop a new search technique that can dramatically improve existing local search methods mentioned above, including the fast TS algorithm of Nowicki and Smutnicki [12]. We present our results as follows. Section II formally defines JSP and outlines the TS algorithm of Nowicki and Smutnicki [12]. Section III presents a new search technique and a new TS algorithm. Section IV gives computational results on an extensive set of test problems. Finally, Section V contains some concluding remarks.

II. Preliminaries

A. Problem Definition

The problem (JSP) is formally defined as follows. There are m *machines* $\mathcal{M} = \{1, \ldots, m\}$ and n *jobs* $\mathcal{J} = \{1, \ldots, n\}$ in a job-shop. Each job $j \in \mathcal{J}$ entails a chain of *operations* $\mathcal{O}_j$ (also known as job-routing constraints); the first operation and the last operation are denoted by f_j and l_j, respectively. Each operation u must be performed on a designated machine $m(u)$, taking $p(u)$ units of time. All machines can perform at most one operation at a time (also known as machine capacity constraints). Let binary relation $\mathcal{A}$ on $\mathcal{O} \equiv \cup_j \mathcal{O}_j$ represent the precedences between operations of the same job, and let binary relation $\mathcal{E}_k$ represent the pairs of distinct operations performed on machine $k \in \mathcal{M}$. Note that $\mathcal{E}_k$ is *symmetric* because $\langle u, v\rangle \in \mathcal{E}_k$ implies that $\langle v, u\rangle \in \mathcal{E}_k$, and so is $\mathcal{E} \equiv \cup_k \mathcal{E}_k$. In addition, let $S(u)$ and $C(u)$ denote the start time and the completion time of operation u, respectively. We try to find a *schedule* specified by start times and completion times of operations. In particular, we are interested in *nonpreemptive* schedules, where $C(u) = S(u) + p(u)$. There are two basic requirements for any schedule: 1) the operation precedences and 2) the machine capacity constraints must be satisfied [17]. With our notation

$$\begin{aligned} \forall \langle u, v\rangle \in \mathcal{A} &: S(v) \geq C(u) \\ \forall k \in \mathcal{M}, \langle u, v\rangle \in \mathcal{E}_k &: S(v) \geq C(u) \\ &\quad \text{or } S(u) \geq C(v). \end{aligned} \tag{1}$$

A schedule that satisfies the constraints in (1) is said to be *feasible*. Let $S_j \equiv S(f_j)$ and let $C_j \equiv C(l_j)$. The classic JSP assumes that $\forall j \in \mathcal{J}, S_j \geq 0$, and seeks for a schedule that minimizes the *makespan*, i.e., the maximum completion time of jobs (denoted by $C_{\max} \equiv \max_j C_j$).

The *disjunctive graph* $\mathcal{G} \equiv (\mathcal{O}, \mathcal{A}, \mathcal{E})$ [18] is a useful tool for visualizing the problem. Node set $\mathcal{O}$ represents the operations, and associated with each node there is a weight equal to the processing time $p(u)$. 2-tuples in binary relations $\mathcal{A}$ and $\mathcal{E}$ maps naturally to (directed) arcs in $\mathcal{G}$. These arcs are divided into two sets: the conjunctive arc set $\mathcal{A}$ and the disjunctive arc set $\mathcal{E}$, depending on which binary relation their corresponding 2-tuples belong to. Due to the symmetry of binary relation $\mathcal{E}$, for each arc in $\mathcal{E}$, the arc of the opposite orientation is also in $\mathcal{E}$. A *partial selection* is a subset of arcs in $\mathcal{E}$ such that at most one in each pair of arcs of opposite orientations is present; it is called a (*complete*) *selection* (denoted by θ) if exactly one arc from each pair is present. A complete (or partial) selection is *feasible* if graph $(\mathcal{O}, \mathcal{A} \cup \theta)$ is acyclic. Minimizing the makespan is equivalent to finding a selection θ so as to minimize the longest path length in the graph $(\mathcal{O}, \mathcal{A} \cup \theta)$. Hence, JSP can be denoted by $P(\theta; \mathcal{O}, \mathcal{A}, \mathcal{E})$, where the selection θ is the decision variable. This notation can be generalized to $P(\theta'; \mathcal{O}, \mathcal{A}, \mathcal{E}')$, where $\mathcal{E}' \subseteq \mathcal{E}$ and the decision variable θ' is a partial selection; $P(\theta'; \mathcal{O}, \mathcal{A}, \mathcal{E}')$ represents a subproblem of JSP that involves only some of the disjunctive arcs in $\mathcal{E}$.

Given a feasible selection θ, the makespan $C_{\max}$ is equal to the length of a longest path (not necessarily unique) in $(\mathcal{O}, \mathcal{A} \cup \theta)$. A longest path is also referred to as a *critical path*. For a given critical path, it can be divided into a number of *blocks*: $B_1, \ldots, B_k$. Each block B_i $(i = 1, \ldots, k)$ is a maximal subset of consecutive operations $\langle a_i, \ldots, b_i\rangle$ processed on the same machine along the critical path; operations a_i and b_i, which satisfy $m(a_i) = \cdots = m(b_i)$ $(i = 1, \ldots, k)$ and $m(b_i) \neq m(a_{i+1})$ $(i = 1, \ldots, k-1)$, are called the *first* and the *last* operations of the block, respectively. Each operation on the critical path is called a *critical operation*. A crucial notion in local search is *neighborhood*, which loosely speaking, is a set of feasible solutions obtained by applying small perturbations to a given feasible solution. Commonly employed small perturbations involve modifying the processing order of certain critical operations.

B. Fast TS Algorithm of Nowicki and Smutnicki

TS is introduced by Glover [19], and it has been applied to JSP (see [9]–[12]). Among the many efficient TS algorithms, we choose the fast TS algorithm of Nowicki and Smutnicki [12] to demonstrate how our proposed technique can be applied.

As one of the local search methods, TS centers around the notion of neighborhood. In particular, Nowicki and Smutnicki [12] employ a very restricted neighborhood. Given a feasible selection θ, denote an arbitrary critical path in graph $(\mathcal{O}, \mathcal{A} \cup \theta)$ by $B_1, \ldots, B_k$, where B_i $(i = 1, \ldots, k)$ are blocks. For each operation u, let $mp(u), ms(u), jp(u)$, and $js(u)$ denote the *machine predecessor* (the operation scheduled immediately before u on the same machine), *machine successor* (the operation scheduled immediately after u on the same machine), *job predecessor* (the operation immediately preceding u in the same job routing), and *job successor* (the operation immediately succeeding u in the same job routing), respectively. The neighborhood $\mathcal{N}_{\rm NS}$ is defined as follows.

Definition 1: $\mathcal{N}_{\rm NS}$ is the set of all the feasible selections obtained in one of the three cases below (for $|B_i| \geq 2$).

- If $i = 1$, reverse the order of $mp(b_i)$ and b_i.
- If $1 < i < k$, reverse the order of $mp(b_i)$ and b_i; moreover, reverse the order of a_i and $ms(a_i)$ if $ms(a_i) \neq mp(b_i)$.
- Otherwise, reverse the order of a_i and $ms(a_i)$.

In TS terminology, the reversal of two operations as mentioned in the above definition is called a *move*. Let π be the current solution and let $V(\pi)$ be the set of all possible moves. During an iteration step, the search makes the transition from π to the next solution by applying a move selected from $V(\pi)$. Taking into account a tabu list T of fixed length $maxt$ (which operates like a cyclic queue), the following two selection rules are used, with the second one being applied only after the first one fails.

- Select the most improving move among moves that either are not in T (i.e., unforbidden) or lead to a smaller makespan than the best thus far (i.e., profitable).

- If $V(\pi)$ contains only one move, select this move; otherwise, select the oldest move v in T (also flush v out of T by appending duplicates of the most recent move to T).

The tabu list T is subsequently updated by appending the inverse of the selected move to T.

The TS algorithm of Nowicki and Smutnicki (referred to as TSAB in [12]) employs the above iteration step in a double loop structure. The inner loop executes this iteration step until the iteration counter exceeds the maximum value *maxiter*. The actual number of executions is affected by two situations. First, whenever a better (or elite) solution π than the incumbent is found, the iteration counter is reset. In addition, three pieces of information are stored in an ordered list L of fixed length $maxl$ (similar to the tabu list), including the elite solution π, the set of remaining moves $V(\pi)\backslash v$ (where v is the move to be applied next), and the current state of the tabu list T. Second, when a cycle is detected, the inner loop terminates immediately.

The outer loop serves the purpose of backtracking, i.e., restarting the inner loop with proper starting values for $\pi, V(\pi)$, and T. When the inner loop is run for the first time, π is found by any heuristic of choice, $V(\pi)$ is the set of moves that induces the neighborhood of π, and $T = \emptyset$ (we also have $L = \emptyset$). Thereafter, the inner loop is restarted with the values of $\pi, V(\pi)$, and T being set to the most recent element in L, provided that $L \neq \emptyset$; otherwise, TSAB terminates. The element in L that has just been referenced is replaced by a new one $(\pi, V(\pi) \setminus v, T)$, where v is the move to be applied next. For additional algorithmic details, we refer the reader to the original paper by Nowicki and Smutnicki.

III. Enhancement Technique for Local Search

A. General Approach

The enhancement technique that we propose is based on the removal and reinsertion of the disjunctive arcs that either emanate from or end at some selected critical operations. Given a schedule, more than one critical paths may exist, and some critical operations may therefore be located on different critical paths. Let $\mathcal{S}_{CP}$ be the set of all critical operations and let $\mathcal{S}_{RR} \subseteq \mathcal{S}_{CP}$ be a selected set of critical operations. Let LS(θ) denote an arbitrary local search method that takes an initial solution (or selection) θ as input and produces a solution that is at least as good as θ. The enhancement technique is outlined as follows.

Algorithm: An Enhanced Local Search

Step 0. Initially, set θ to a feasible solution generated by any constructive method.

Step 1. Improve θ by calling LS. If no improvement is found, *STOP*. Otherwise, let θ' denote the improved solution.

Step 2. Given θ', set $\mathcal{S}_{RR}$ to a subset of $\mathcal{S}_{CP}$.

Step 3. Relax the machine capacity constraint for each operation in $\mathcal{S}_{RR}$; i.e., remove all the disjunctive arcs $\mathcal{E}_{RR}$ that involve the operations in $\mathcal{S}_{RR}$ from the disjunctive graph $(\mathcal{O}, \mathcal{A}, \mathcal{E})$, resulting in a disjunctive subgraph $(\mathcal{O}, \mathcal{A}, \mathcal{E}\backslash\mathcal{E}_{RR})$ that retains all the original nodes. Apply a modified version of LS (see later) to the disjunctive subgraph and let the solution found be denoted by θ''. Note that θ'' is a partial selection to JSP.

Step 4. In a certain order, reimpose the machine capacity constraint on each operation in $\mathcal{S}_{RR}$, one at a time. Suppose that an operation $o \in \mathcal{S}_{RR}$ needs to be processed on machine $m(o)$ and that k operations have already been sequenced on machine $m(o)$. While preserving the precedences between the existing operations on machine $m(o)$, insert o into one of the $k+1$ positions (before, between, or after those k operations) such that the increase in the makespan is minimal. (See [12] for more details on the insertion.)

Step 5. Let θ''' be the new solution. Set θ to θ''' and go to 1.

Step 3 calls for a modified version of LS for optimizing the processing orders on machines after the disjunctive arcs in $\mathcal{E}_{RR}$ is removed from the disjunctive graph. Note that the operations in $\mathcal{S}_{RR}$ still participate in the problem except that they do not occupy machines and therefore can be regarded as "delays". Hence, the minimum makespan of the sequencing problem on the disjunctive subgraph $(\mathcal{O}, \mathcal{A}, \mathcal{E}\backslash\mathcal{E}_{RR})$ is a lower bound on the minimum makespan of the original problem.

To apply the enhancement technique to TS, we modify the algorithm of Nowicki and Smutnicki to deal with "delays", which is described next.

B. Implementation of the Algorithm of Nowicki and Smutnicki

We implement the TS algorithm proposed by Nowicki and Smutnicki [12] with a couple of modifications. We shall refer to our algorithm as TS and the one by Nowicki and Smutnicki as TSAB. Our algorithm TS uses the same neighborhood structure, tabu tenure (tabu list size), criteria for selecting the next move among unforbidden (U), forbidden but profitable (FP), forbidden and nonprofitable (FN) moves, and backtracking scheme. Unlike TSAB in [12], the cycle detection in TS utilizes a hash table; a cycle is declared and a backtracking subsequently occurs after any particular makespan value repeats itself for more than 1000 times. In our algorithm, the cycle detection mechanism is reset only during backtracking, whereas in TSAB, it is reset either when a backtracking occurs or when a makespan better than the currently best is found. By using the hash table, TS achieves comparable solution quality as TSAB but requires less computation time. Furthermore, TS explicitly handles "delays" (i.e., delay operations) as required by our proposed enhancement technique.

Delay operations can be treated in a natural fashion. Suppose that an operation u is the only delay operation (more than one delay operations are handled similarly). Recall that the TS algorithm solves problem $P(\theta; \mathcal{O}, \mathcal{A}, \mathcal{E})$ (Section II-A). Now, we need to solve $P(\theta'; \mathcal{O}, \mathcal{A}, \mathcal{E}')$, where $\mathcal{E}'$ is obtained by removing all the arcs in $\mathcal{E}$ that either emanate from or end at u. We may regard u as being performed on a distinct fictive machine, and clearly, no sequencing decisions need to be made for u. Note that the resulting problem can be viewed as another instance of JSP with one more machine. Therefore, any local search method LS can be easily modified to account for delay operations. Varying the choice of LS yields various enhanced local search methods.

With regard to our TS algorithm, if u happens to be on the critical path, it must be in a block containing only a single operation (just u itself). By the definition of the neighborhood $\mathcal{N}_{\rm NS}$, such a block will not contribute a neighbor. Based on this observation, the TS algorithm is modified accordingly. Note that TS behaves the same way as TSAB when there are no delay operations.

The parameter setting of TS is specified as follows. The tabu tenure $maxt$ is set to 8 and the backtracking depth $maxl$ is set to 5. The maximum iteration limit max*iter* is set to 3000 initially and if the currently best makespan is improved during the run. The *maxiter* is set to $3000 - 500\ (maxl - l)$ during backtracking, where l is the number of recorded elite solutions.

C. A New TS Algorithm

Now we apply our general approach described in Section III-A to TS to obtain a new TS algorithm, which is referred to as TSEn. After some experiments, we find that the following configuration works well.

- The initial solution is generated using INSA, which is an insertion heuristic described in [12].
- In Step 1, TS is applied.

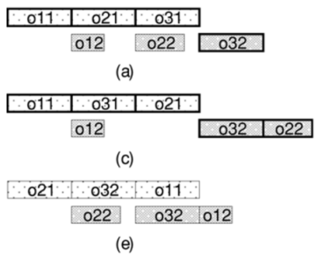

Fig. 1. Breakdown of connectivity property. (a) $C_{\max} = 4$. (b) $C_{\max} = 4.75$. (c) $C_{\max} = 4.75$. (d) $C_{\max} = 4.75$. (e) $C^*_{\max} = 3.5$. (f) $C^*_{\max} = 3.5$.

- In Step 2, $\mathcal{S}_{RR} = \{a_i, b_i \mid i = 1, \ldots, k\}$, which contains the first and the last operations of blocks. Note that a_1 and b_k are included in $\mathcal{S}_{RR}$ for diversification purposes, although they are not involved in the definition of $\mathcal{N}_{\rm NS}$ when their respective block sizes are greater than 2.
- In Step 3, TS is applied after the operations in $\mathcal{S}_{RR}$ are labeled as "delays".
- In Step 4, the machine capacity constraint is reimposed on each operation in $\mathcal{S}_{RR}$ in ascending order of earliest start times as implied by θ'.
- We use the following (greedy) stopping criterion: Stop if the currently best makespan is not improved during an iteration.

The proposed procedure TSEn embodies the idea of strategic oscillation suggested by Glover and Laguna [19]. The procedure alternates optimization phases between the solution space of the original problem and those of the relaxed problems (infeasible solutions). This provides an effective diversification mechanism, which helps the search explore regions of the solution space that may be difficult to reach with local search.

Before discussing detailed computational experiments, we show the working of the proposed enhancement scheme through a simple example (the construction is similar to the one given in [9]). As illustrated in Fig. 1, we have a problem instance with three jobs $(j = 1, 2, 3)$, each of which requires two successive operations (o_{j1} and o_{j2}) on machines 1 and 2, respectively. All the operations take 1 unit of time except the second operations of Job 1 (i.e., o_{12}) and Job 2 (i.e., o_{22}), which take 0.5 and 0.75 unit of time, respectively. Fig. 1(a) shows an initial solution. Fig. 1(b)–(d) are the only solutions reachable from Fig. 1(a) by the TS algorithm. Therefore, neither of the two optimal solutions shown in Fig. 1(e) and (f) can be reached from the given initial solution. This failure to attain optimality by TS is caused by the fact that the definition of the neighborhood $\mathcal{N}_{\rm NS}$ does not satisfy the connectivity property, which has been acknowledged in [12].

The breakdown of the connectivity property as with the neighborhood structure $\mathcal{N}_{\rm NS}$ is undesirable but often inevitable. Without connectivity, the quality of the best solution that a local search method is able to find is likely to be sensitive to the initial solution. On the other hand, effective local search methods for JSP such as TS favors small neighborhoods, since having fewer solutions in a neighborhood allows the methods to run faster; however, it is generally more difficult to retain connectivity with smaller neighborhoods. What Fig. 1 shows is an example of the entrapment of local search methods by local optima when the connectivity property no longer holds. Our proposed enhancement technique tries to remedy the situation.

Fig. 2 shows how the enhancement technique helps guide the search out of the local optimum as depicted by Fig. 1(a). According to the enhanced algorithm TSEn, operations o_{11}, o_{31}, and o_{32} in this local optimum are labeled as delay operations on fictive machines $M3, M4$, and $M5$, respectively. Fig. 2(a) corresponds to the partial selection θ''. Since operations o_{11}, o_{31}, and o_{32} are already in ascending order of earliest start times, one at a time the machine capacity constraints are reimposed on them in such a way that causes the smallest increase in the makespan in each step (see Fig. 2(b)–(d)). Starting from the solution depicted in Fig. 2(d), TS finds the optimal solution in Fig. 1(f) in two iterations.

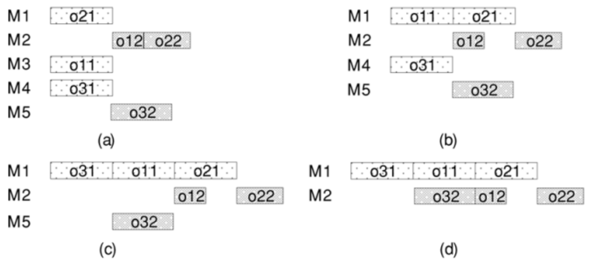

Fig. 2. Result of applying TSEn. (a) $C_{\max} = 2.25$, (b) $C_{\max} = 2.75$, (c) $C_{\max} = 3.75$, (d) $C_{\max} = 3.75$.

TABLE I
SUMMARY OF COMPUTATIONAL RESULTS BY MRTS AND TSEn FOR 242 INSTANCES

Problem	No. outperform*			Mean RG (%)	
	Tie	MRTS	TSEn	MRTS	TSEn
FT6,10,20	3	0	0	0.00	0.00
LA1-40	34	3	3	0.23	0.23
ABZ5-9	2	0	3	1.40	1.26
ORB1-10	5	1	4	0.49	0.27
SWV1-20	7	3	10	2.39	1.79
YN1-4	1	0	3	1.29	0.83
TD1-80	38	25	17	1.18	1.25
DMU1-80	14	18	48	1.93	1.22

* number of times that one algorithm outperforms the other

IV. Computational Results

We coded the new TS algorithm (TSEn) in Visual C++ on a Pentium III 733 MHz PC. For comparison purposes, we also coded a multirestart TS algorithm (MRTS) that restarts TS with different initial solutions. In this multirestart algorithm, the initial solution for the first run of TS is generated using INSA (the insertion heuristic as mentioned before), and initial solutions for subsequent runs are generated in turn using ten of the most effective priority dispatching heuristics in the literature (see [20], [21]) and then their randomized versions. These ten dispatching heuristics are applied in the same order as they are listed in [20]. For each test problem, the time limit for MRTS is set to the same amount of time as what is required for TSEn to terminate.

The test problems that we used are 242 well-known benchmark problems for JSP contributed by Fisher and Thompson [1] and Lawrence [2], among other authors (see [22] for a complete list). These problems are FT6, FT10, and FT20, LA1-40, ORB1-10, SWV1-20, YN1-4, TD1-80, and DMU1-80. For TD instances, the best known upper bounds $({\rm UB}_{\rm best})$ and the best known lower bounds $({\rm LB}_{\rm best})$ are taken from [23]. For the remaining instances, ${\rm UB}_{\rm best}$ and ${\rm LB}_{\rm best}$ are taken from [22] and updated with the improved results from [24]. 111 of the 242 instances have previously been solved to optimality, and for those unsolved instances, the mean relative deviation between the best known upper bound and the best lower bound is 7.72%. To evaluate the quality of our solutions, we compare their associated makespans (UB) with the upper bounds and compute the relative gaps, i.e., ${\rm RG} = ({\rm UB} - {\rm UB}_{\rm best})/{\rm UB}_{\rm best}$.

We applied MRTS and TSEn to the test problems. In Table I, the three columns under the title "No. outperform" show the numbers of instances in each problem set for which both algorithms obtained the same objective value, for which MRTS outperformed TSEn, and for which TSEn outperformed MRTS, respectively. In the same table, the mean relative gap is also reported for each problem set. (Detailed results for all the 242 problems are available online at [25].) These results

TABLE II
IMPROVED UPPER BOUNDS FOR DMU INSTANCES

Problem	n	m	UB_{best}	MRTS	TSEn		
					UB	RG (%)	CPU (sec.)
DMU6	20	20	3269	3332	3254	-0.46	365
DMU10	20	20	3001	3000	2994	-0.23	78
DMU11	30	15	3491	3533	3484	-0.20	138
DMU20	30	20	3788	3792	3757	-0.82	540
DMU27	40	20	4883	4873	4871	-0.25	644
DMU42	20	15	3574	3467	3544	-0.84	216
DMU48	20	20	3918	3968	3894	-0.61	338
DMU56	30	20	5234	5394	5177	-1.09	1270
DMU58	30	20	5038	5118	4937	-2.00	748
DMU64	40	15	5580	5732	5553	-0.48	696
DMU72	50	15	6790	7220	6761	-0.43	1215

indicate that the proposed enhanced TS algorithm TSEn and MRTS yield the same solution quality on problem sets FT and LA, which are two relatively easy sets of problems. TSEn outperforms MRTS on all the other problem sets exception TD, where TSEn is slightly worse than MRTS. Overall, TSEn is superior to MRTS. Furthermore, TSEn is able to improve the best known upper bounds $(\mathrm{UB}_{\mathrm{best}})$ for 11 instances in problem set DMU, which contains a larger number of difficult instances than any other problem set. By contrast, MRTS is only able to improve $\mathrm{UB}_{\mathrm{best}}$ for 3 DMU instances, all of which happen to be among those 11 instances. For those 11 DMU instances, Table II reports the improved upper bound values and the CPU times (in seconds) required by TSEn to reach these values, as well as the objective values attained by MRTS. Table II also indicates that TSEn produces better results than MRTS for the 11 instances, with the exception of DMU42 (on this particular occasion, MRTS gives a much better solution).

V. CONCLUSION

In this short paper, we developed an easy-to-implement enhancement technique for local search methods. Our idea is illustrated by applying the technique to a TS algorithm proposed by Nowicki and Smutnicki [12]. However, our method can be applied to other local search methods as well. Extensive numerical results have shown that the proposed technique, despite its simplicity, is an effective way of realizing the tradeoff between CPU time and solution quality.

Local search methods seem to bear some hope for solving large instances of the JSP. These methods can approach a good solution quickly while having some ability to escape the entrapment by local optima. However, additional techniques such as the one proposed here are needed to help further remedy the myopic nature of local search.

ACKNOWLEDGMENT

We would like to thank the associate editor and two anonymous referees for their helpful comments and suggestions, which have helped strengthen the content and presentation of this paper.

REFERENCES

[1] H. Fisher and G. L. Thompson, "Optical reflectometry for component characterization," in *Industrial Scheduling*, J. F. Muth and G. L. Thompson, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1963.

[2] S. Lawrence, *Supplement to Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques*. Pittsburgh, PA: GSIA, Carnegie-Mellon Univ., 1984.

[3] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Manag. Sci.*, vol. 32, pp. 391–401, 1988.

[4] E. Balas, J. K. Lenstra, and A. Vazacopoulos, "The one-machine problem with delayed precedence constraints and its use in job shop scheduling," *Manag. Sci.*, vol. 41, pp. 94–109, 1995.

[5] R. J. M. Vaessens, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by local search," *INFORMS J. Comput.*, vol. 8, pp. 302–317, 1996.

[6] H. Matsuo, C. J. Suh, and R. S. Sullivan, *A Controlled Search Simulated Annealing Method for the General Job-Shop Scheduling Problem*. Austin, TX: Grad. School of Bus., Univ. of Texas, 1988.

[7] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Oper. Res.*, vol. 40, pp. 113–125, 1992.

[8] E. H. L. Aarts, P. van Laarhoven, J. K. Lenstra, and N. L. J. Ulder, "A computational study of local search algorithms for job shop scheduling," *ORSA J. Comput.*, vol. 6, pp. 118–125, 1994.

[9] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Ann. Oper. Res.*, vol. 41, pp. 231–252, 1993.

[10] E. Taillard, "Parallel taboo search techniques for the job-shop scheduling problem," *ORSA J. Comput.*, vol. 16, pp. 108–117, 1994.

[11] J. W. Barnes and J. B. Chambers, "Solving the job shop scheduling problem using tabu search," *IIE Trans.*, vol. 27, pp. 257–263, 1995.

[12] E. Nowicki and C. Smutnicki, "A fast tabu search algorithm for job shop problem," *Manag. Sci.*, vol. 42, pp. 797–813, 1996.

[13] T. Yamada and R. Nakano, "A genetic algorithm applicable to large-scale job-shop problems," in *Proc. 2nd Int. Workshop on Parallel Problem Solving from Nature*, R. Männer and B. Manderick, Eds., Brussels, Belgium, pp. 281–290.

[14] F. Della Croce, R. Tadei, and G. Volta, "A genetic algorithm for the job shop problem," *Comp. Oper. Res.*, vol. 22, pp. 15–24, 1995.

[15] U. Dorndorf and E. Pesch, "Evolution based learning in a job-shop scheduling environment," *Comp. Oper. Res.*, vol. 22, pp. 25–40, 1995.

[16] E. Balas and A. Vazacopoulos, "Guided local search with shifting bottleneck for job-shop scheduling," *Manag. Sci.*, vol. 44, pp. 262–275, 1998.

[17] E. Balas, "Disjunctive programming," *Ann. Discrete Math.*, vol. 5, pp. 3–51, 1979.

[18] B. Roy and B. Sussmann, *Les problèmes d'ordonnancements avec contraintes disjonctives*. Paris, France: SEMA, 1964.

[19] F. Glove and M. Laguna, *Tabu Search*. Norwell, MA: Kluwer, 1997.

[20] A. S. Jain, B. Rangaswamy, and S. Meeran, "New and 'stronger' job shop neighborhoods: A focus on the method of Nowicki and Smutnicki," *J. Heur.*, vol. 6, pp. 457–480, 2000.

[21] Y. L. Chang, T. Sueyoshi, and R. S. Sullivan, "Ranking dispatching rules by data envelopment analysis in a job-shop environment," *IIE Trans.*, vol. 28, pp. 631–642, 1996.

[22] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present, and future," *Eur. J. Oper. Res.*, vol. 113, pp. 390–434, 1999.

[23] É. Taillard. (2002) Job shop test problems. [Online]. Available: http://www.eivd.ch/ina/collaborateurs/etd/problemes.dir/ordonnancement.dir/best_lb_up.txt

[24] W. Brinkkötter and P. Brucker, "Solving open benchmark instances for the job-shop problem by parallel head-tail adjustments," *J. Sched.*, vol. 4, pp. 53–64, 2001.

[25] L. Shi and Y. Pan. Detailed computational results for "An efficient search method for job-shop scheduling problems". [Online]. Available: http://www.cae.wisc.edu/pany/enhanced_tabu/table.pdf