Compact Local Search Neighborhoods For Generalized Scheduling Problems

Selcuk Avci
marketRx, Inc.

Robert H. Storer
Lehigh University

Report No. 04T-001

Compact Local Search Neighborhoods For Generalized Scheduling Problems

Selcuk Avci
marketRx, Inc.

Robert H. Storer
Lehigh University

Report No. 04T-001

Compact Local Search Neighborhoods for Generalized Scheduling Problems

Selcuk Avci[2] and Robert H. Storer[1,*]

Abstract

In this study we develop effective local search neighborhoods for a broad class of scheduling problems. We prove many non-improving moves in adjacent pairwise interchange neighborhoods can be eliminated while also addressing move feasibility and quality issues. In our theoretical development we use a quite general problem definition that encompasses a variety of well-known regular and non-regular scheduling problems including job shop problems with ready times, distinct due dates, and weighted penalties for tardiness and earliness. We show that our compact neighborhood definition generalizes Nowicki and Smutnicki's neighborhood for the classical job shop (Nowicki and Smutnicki, 1996) to our broader class of problems. We note that Nowicki and Smutnicki's neighborhood is the most effective search neighborhood in the literature for the classical job shop problem and has been instrumental in the development of powerful solution methods. The work presented in this paper discovers an important link between regular and non-regular scheduling problems and provides a direct method to extend effective, known local search algorithms to non-regular scheduling problems.

1. Introduction

In this paper we develop effective local search neighborhoods for a quite general scheduling problem representation. We show that our neighborhood results significantly generalize previous important results for classical (makespan) job shop problems. A notable feature is that the results apply to non-regular scheduling problems. The vast majority of scheduling research assumes objective functions that are non-decreasing in job completion times. Among these "*regular* performance measure problems", objective functions with tardiness terms are often used to measure conformance to due dates. These performance measures ignore job earliness, although in many realistic cases earliness penalties are significant. Here we use the new neighborhood results to develop algorithms for *non-regular* generalized job shop scheduling problems with earliness and tardiness penalties. We further generalize by addressing the problem at the operation level (where early-tardy penalties may be associated with each operation) rather than at the job level that is customary in scheduling theory. This allows us to penalize work-in-process inventory in addition to finished goods.

Earliness penalties arise when inventory must be carried until delivery, and/or when cash is committed earlier than needed. Tardiness penalties arise from loss of customer goodwill and future sales, or direct cash penalties. Further justification is derived from the emergence of the Just-in-Time (JIT) philosophy that views both earliness and tardiness as costly. Rescheduling is another important use of non-regular early-tardy scheduling. Suppose a detailed production schedule exists, and that all the necessary materials and resources have been allocated. When a disturbance occurs, one must find a

[1]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, USA
[2]marketRx Inc., Bridgewater, NJ 08807, USA
[*]Corresponding author

new schedule that follows the original as closely as possible. Such a schedule would try to minimize materials reallocation, and limit the impact of schedule changes at higher levels of the production planning hierarchy. To accomplish this, every operation in the "rescheduling problem" would have distinct release times and due dates imposed by the pre-existing schedule.

2. Literature Survey

Baker and Scudder (1990) and Kanet and Sridharan (2000) provide comprehensive reviews of the literature on deterministic non-regular scheduling problems with earliness and tardiness penalties (often referred to as "E/T problems"). These reviews indicate that the vast majority of the existing E/T literature deals with deterministic static single-machine problems. Some studies have extended single machine results to parallel machines, and a very few address dynamic single-machine problems. Further, most existing E/T studies avoid the issue of deliberate idle time insertions either by directly prohibiting them or by assuming a common due date for all jobs thus allowing them to be *treated as* regular scheduling problems. Although these reviews were published a decade apart we observe little progress in the research on multi-machine E/T problems during 1990s. According to Kanet and Sridharans's taxonomy, multi-machine non-regular scheduling problems with distinct release times are the most general class of inserted idle time problems. The few studies addressing these problems are summarized next.

Faaland and Schmitt (1987) address a general E/T problem with multiple machines and product structures. They assume an earliness penalty for every task of all products, however the tardiness penalty is incurred only by the end products. They approach the problem in two stages by first heuristically assigning the sequence of operations at each work center so that product structure constraints are satisfied. In the second stage they solve the idle time assignment problem by iteratively solving an associated maximum flow problem. This study considers the generation of a feasible schedule only, and does not address the sequencing issue beyond the single sequence generated heuristically in stage 1.

Storer et al. (1992) provide a problem space genetic algorithm for the problem of minimizing average absolute deviation from due dates in the job shop setting. This problem is static with distinct due dates for the last operation of each job. They use a two-stage base heuristic to generate schedules with idle times. The first stage uses the minimum slack dispatching rule to generate a sequence, while the second stage inserts idle times into this sequence heuristically. The two-stage base heuristic is then embedded in a Problem Space search algorithm for further improvement. Their computational tests obtained very promising results.

Wennink (1995) concentrates on the design of a framework for an automated planning board generator (PBG) for complex planning problems. He introduces different aspects of PBG design including the user interface, problem representation, and problem manipulation. He then focuses on the algorithmic support for problem manipulation in the context of deterministic machine scheduling problems. The author proposes a generalization of the classical job shop scheduling problem in both constraints and objective function. In particular this general scheduling problem includes release times, deadlines, minimum and maximum delay constraints, non-regular and multi-criteria objectives. Wennink (1995) shows that the quality (objective function value) of a schedule can be determined by solving a maximum cost network flow problem and uses this relationship to extend some of the solution methods for classical job shop problem so that they can also be applied to the general scheduling problem. Several solutions methods have been studied in detail including constructive solution methods, local search, and Lagrangian relaxation. In particular the author presents general adjacent pairwise interchange and insertion heuristics for different subclasses of the general scheduling problem. In principle, these methods may handle problems with certain non-regular objectives. However specific solution methods have been developed and tested only for regular scheduling problems with the makespan objective.

Brandimarte and Maiocco (1999) study job shop scheduling with a non-regular objective function that penalizes both tardiness and holding time. Holding time for a job is defined as the time between the start of its first operation and its delivery. The authors claim that an explicit earliness term does not capture the multi-operation nature of their problem. However, their objective function accounts for earliness indirectly through the holding cost term. They give a decomposition framework based on the two-stage approach that is commonly used for non-regular scheduling problems (i.e. sequencing followed by idle time insertion or "timing"). The authors propose a maximum cost network flow representation for their timing sub-problem. They use this representation to prove certain properties of adjacent pairwise interchange neighborhoods. Because of the computational requirement of solving a maximum cost network flow problem the authors also propose a heuristic. The authors employ a variety of job release and dispatching rules to construct an initial solution, then use basic local search on adjacent pairwise interchange neighborhoods for further improvement. The authors also underlined the need for a more compact search neighborhood similar to Nowicki and Smutnicki's adjacent pairwise interchange neighborhood for the classical job shop scheduling problem (Nowicki and Smutnicki, 1996).

Both Wennink (1995) and Brandimarte and Maiocco (1999) investigate adjacent pairwise interchange neighborhoods for their scheduling problems. In this study we reproduce some of their results, and more importantly, we provide further theoretical development that leads to more compact search neighborhoods for a variety of scheduling problems. We also discuss the relationships between our neighborhood and other studies in the scheduling literature.

3. Problem Statement and Representations

We assume a set J of n jobs to be processed on a set M of m machines in a job shop setting. We make the following common assumptions:

- No preemption: once an operation is begun on a machine, it cannot be interrupted.
- Each machine can process only one operation at a time.
- An operation of a job cannot begin until its predecessor operation of the same job is complete.
- Each job visits each machine exactly once (this assumption is not critical to any of the methods we develop.)

Let O_j be the set of operations for job j and let $O = \bigcup_{j \in J} O_j$ denote the set of all operations.

Associated with each operation $i \in O$ are the following nonnegative values: processing time p_i, release time r_i, due date d_i, earliness penalty α_i, and tardiness penalty β_i. Note that assigning release times, due dates and penalties to each operation rather than to the each job generalizes many common formulations. We further include a nonnegative makespan penalty γ assigned to the schedule as a whole. We use this makespan penalty term to establish relationships between our results for non-regular scheduling problem and previous results for the classical job shop problem.

There is a unique machine $m(i) \in M$ required to process each operation $i \in O$. Further let $o(j,k) \in O_j$ denote the k^{th} operation of job j, and let $jp(i)$ and $js(i)$ denote the preceding and succeeding operations of the same job for operation i. We define the set of job related precedence relationships, F, among the operations as: $F = \{(o(j,k), o(j,k+1)) \mid k = 1,\cdots,m-1 \text{ and } j \in J\}$. We also define a set of start-operations, O^S, and a set of end-operations, O^E such that: $O^S = \{o(j,1) | j \in J\}$ and $O^E = \{o(j,m) | j \in J\}$.

A Disjunctive Formulation

Our generalized scheduling problem (GSP) is to find a starting time t_i for each operation $i \in O$ minimizing the total weighted earliness and tardiness and makespan cost:

$$\text{Minimize} \quad Z_{GSP} = \sum_{i \in O} (\alpha_i \cdot x_i + \beta_i \cdot y_i) + \gamma \cdot z$$

$$\text{s.t:} \quad t_j \geq t_i + p_i \qquad \text{for } (i,j) \in F \qquad (1)$$

$$t_j \geq t_i + p_i \vee t_i \geq t_j + p_j \qquad \text{for } \{i,j\} \in \bigcup_{k \in M} P_k^M \qquad (2)$$

$$z \geq t_i + p_i \qquad \text{for } i \in O^E \qquad (3)$$

$$t_i \geq r_i \qquad \text{for } i \in O \qquad (4)$$

$$t_i + p_i = d_i + y_i - x_i \qquad \text{for } i \in O \qquad (5)$$

$$x_i, y_i \geq 0 \text{ and } t_i \text{ free} \qquad \text{for } i \in O$$

$$z \text{ free}$$

where,

P_k^M : set of unordered operation pairs sharing the same machine k, i.e.

$$P_k^M = \{\{i,j\} \mid m(i) = m(j) = k \text{ and } i < j \text{ for } i,j \in O\}$$

x_i : earliness of operation $i \in O$, i.e. $x_i = \max\{d_i - (t_i + p_i), 0\}$

y_i : tardiness of operation $i \in O$, i.e. $y_i = \max\{(t_i + p_i) - d_i, 0\}$

z : makespan of the schedule, i.e. $z = \max_{i \in O^E}\{t_i + p_i\}$

Constraint set (1) separates consecutive operations of a job in time. Similarly, disjunctive constraint set (2) ensures separation among operations sharing the same machine. Constraint set (3) ensures that makespan is at least as large as the completion time of any end-operation. Constraint set (4) imposes the release time constraints. Finally, constraint set (5) defines earliness x_i and tardiness y_i for each operation i as a function of t_i.

In this generalized problem we allow E/T penalties for all operations rather than only the end-operations. This feature is particularly important for representing *rescheduling* problems with problem (GSP), as discussed in Section 1. Further, problem (GSP) covers a wide variety of both regular and non-regular scheduling problems including the well-known classical (makespan) job shop problem, the total weighted tardiness job shop problem, and several single-machine problems. Problem (GSP) is *NP*-Hard since many of its special cases (for example, the single-machine total weighted tardiness problem (Du and Leung, 1990), have been shown to be *NP*-Hard.

The solution of problem GSP entails selecting an orientation for each disjunctive relationship, i.e. convert them into conjunctive relationships. These conjunctive relationships are precedence relationships among the operations that share a common machine, thus we will refer to them as *machine precedence relationships*. We use $i \prec j$ to denote that operation j

cannot begin until operation i is complete. If $i \prec j$ and there is no operation k such that $i \prec k$ and $k \prec j$ then this is an *immediate precedence relationship.*

For a given operation sequence σ we can define a *maximal* precedence relationship set as that which includes all possible machine precedence relationships dictated by the transitive property. We can also define a *minimal* precedence relationship set with respect to a given operation sequence σ which includes only the immediate machine precedence relationships for consecutive operations. Let S_k be a set of precedence relationships for machine k. We define the set of precedence relationships, S, across all machines as $S = \bigcup_{k \in M} S_k$. A machine precedence relationship set S is defined to be a *complete* if S maps into a complete operation sequence σ, otherwise S is a *partial* machine precedence relationship set. The chains of precedence relationships from *the start-operation* to *the end-operation* for each job $j \in J$ are referred to as *job precedence relationships*. Let F denote the set of all job precedence arcs. The complete set of precedence relationships, P, is defined as $P = F \cup S$. The precedence relationships in P are feasible for problem (GSP) if and only if the corresponding digraph $G(O,P)$ is acyclic. We will call these types of precedence relationships *feasible* precedence relationships.

An Alternative Problem Representation

Several studies in the literature suggest a two-stage solution method for non-regular scheduling problems (Faaland and Schmitt (1987), Fry et al. (1987), Yano and Kim (1991), Storer et al. (1992) and Fry et al (1996)). In these solution methods the first stage involves finding a feasible operation sequence and possibly a regular semiactive schedule. In the second stage idle times are inserted to construct a good schedule associated given the stage one operation sequence. By embedding the stage 2 "optimal timing problem" (OTP) we provide the following alternative formulation for problem (GSP) defined on the operation sequence space:

$$Z_{GSP} = \min_{\sigma \in \Omega} \{Z_{OTP}(\sigma)\}$$

where Ω is the set of all feasible operation sequences. Note that this formulation is closely related to the disjunctive graph model given in the previous section. A feasible operation sequence σ defines a feasible selection of orientations for disjunctive edges in a disjunctive graph. Given a sequence σ, Z_{GSP} becomes a MCNF problem as has been observed by Brandimarte and Maiocco (1999) and Wennink (1995).

The above representation of the generalized scheduling problem provides some insight for developing heuristic methods. We can relate each sequence to its optimal non-regular semiactive schedule via problem (OTP). Thus as is typical for regular scheduling problems, we may view the problem as finding feasible sequences that result in good schedules.

The Optimal Timing Problem

In this section we consider the problem of finding optimal start times of operations for a given set of precedence relationships. In the literature, the optimal timing problem has also been referred to as the idle time insertion problem (Fry et al., 1996) and the timetabling problem (Davis and Kanet, 1993). Let S be a machine precedence relationship (which is at least *minimal*) associated with a complete operation sequence σ.

Problem OTP:

$$\text{Minimize} \quad Z_{OTP}(\sigma) = \sum_{i \in O} (\alpha_i \cdot x_i + \beta_i \cdot y_i) + \gamma \cdot z$$

$$\begin{aligned}
\text{s.t:} \quad & t_j \geq t_i + p_i && \text{for } (i,j) \in \boldsymbol{P} \\
& z \geq t_i + p_i && \text{for } i \in \boldsymbol{O}^E \\
& t_i \geq r_i && \text{for } i \in \boldsymbol{O} \\
& t_i + p_i = d_i + y_i - x_i && \text{for } i \in \boldsymbol{O} \\
& x_i, y_i \geq 0 \text{ and } t_i \text{ free} && \text{for } i \in \boldsymbol{O} \\
& z \text{ free}
\end{aligned}$$

where $\boldsymbol{P}$ is set of all precedence relationships, i.e. $\boldsymbol{P} = \boldsymbol{F} \cup \boldsymbol{S}$. This LP formulation can be transformed into a maximum cost network flow problem and thus solved more quickly than general LP problems as we now show.

A Maximum Cost Network Flow Model

We begin the transformation by defining lateness π_i as a free variable in terms of earliness and tardiness,

$$\pi_i = t_i + p_i - d_i = y_i - x_i$$

Next we redefine the starting times t_i as

$$t_i = (d_i - p_i) + \pi_i$$

We also define the following constants:

$$c_{i,j} = d_i - (d_j - p_j)$$

$$c_{r,j} = r_i - (d_i - p_i)$$

$$c_{i,z} = d_i$$

Using the above definitions we transform problem (OTP) into problem (P) as follows:

$$\text{Minimize} \quad Z_P(\boldsymbol{S}, \pi) = \sum_{i \in O} (\alpha_i \cdot x_i + \beta_i \cdot y_i) + \gamma \cdot z$$

s.t:

$$\begin{aligned}
[f_{i,j}] \quad & -\pi_i + \pi_j \geq c_{i,j} && \text{for } (i,j) \in \boldsymbol{P} \\
[f_{i,z}] \quad & z - \pi_i \geq c_{i,z} && \text{for } i \in \boldsymbol{O}^E \\
[f_{r,i}] \quad & \pi_i \geq c_{r,i} && \text{for } i \in \boldsymbol{O} \\
[f_{s,i}] \quad & x_i \geq 0 && \text{for } i \in \boldsymbol{O} \\
[f_{i,t}] \quad & y_i \geq 0 && \text{for } i \in \boldsymbol{O}
\end{aligned}$$

π_i free $\qquad\qquad\qquad\qquad$ for $i \in \boldsymbol{O}$

The dual of the above formulation has a network structure where set O becomes a node set, and set $\boldsymbol{P}$ becomes an arc set defined over $\boldsymbol{O}$.

We transform the dual of problem (OTP) into a circulatory Maximum Cost Network Flow (MCNF) problem by defining a directed graph $G(V, A \cup P)$ where

$$V = O \cup \{s,t,r,z\}$$

$$A = \left(\bigcup_{i\in O}\{(r,i),(s,i),(i,t)\}\right) \cup \left(\bigcup_{i\in O^E}\{(i,z)\}\right) \cup \{(t,s),(s,r),(z,s)\}$$

In this graph, the node set contains the set of operations along with a source node *s*, a sink node *t*, a release node *r* and a fictitious terminal operation node *z*. The set of arcs contains the precedence relationship set $\boldsymbol{O}$, and a set of three arcs (s,i), (r,i) and (i,t) joining operation *i* to node *s*, *r* and *t* for each operation $i \in \boldsymbol{O}$. For each end-operation $i \in \boldsymbol{O}^E$ we have an arc (i,z). In addition, arcs (t,s), (s,r) and (z,s) are also included to achieve a balance of flow at nodes *r*, *s*, *t*, and *z* so that the problem is a *circulatory* MCNF problem.

Now, we can formulate problem (MCNF) for $G(V, A \cup P)$ as follows:

Maximize $\quad Z_D(S,f) = \sum_{(i,j)\in P} c_{i,j}\cdot f_{i,j} + \sum_{i\in O} c_{r,i}\cdot f_{r,i} + \sum_{i\in O^E} c_{i,z}\cdot f_{i,z}$

s.t:

$[\pi_i] \quad -\sum_{(i,j)\in P} f_{i,j} + \sum_{(j,i)\in P} f_{j,i} + f_{r,i} + f_{s,i} - f_{i,t} = 0 \qquad$ for $i \in \boldsymbol{O}\setminus\boldsymbol{O}^E$

$[\pi_i] \quad -\sum_{(i,j)\in P} f_{i,j} + \sum_{(j,i)\in P} f_{j,i} + f_{r,i} + f_{s,i} - f_{i,t} - f_{i,z} = 0 \qquad$ for $i \in \boldsymbol{O}^E$

$[\pi_r] \quad f_{s,r} - \sum_{i\in O} f_{r,i} = 0 \qquad$ for *r*

$[\pi_s] \quad f_{t,s} + f_{z,s} - \sum_{i\in O} f_{s,i} - f_{s,r} = 0 \qquad$ for *s*

$[\pi_t] \quad \sum_{i\in O} f_{i,t} - f_{t,s} = 0 \qquad$ for *t*

$[\pi_z] \quad \sum_{i\in O^E} f_{i,z} - f_{z,s} = 0 \qquad$ for *z*

$[x_i] \quad f_{s,i} \le \alpha_i \qquad$ for $i \in \boldsymbol{O}$

$[y_i] \quad f_{i,t} \le \beta_i \qquad$ for $i \in \boldsymbol{O}$

$[z] \quad f_{z,s} \le \gamma$

$\quad f_{i,j} \ge 0 \qquad$ for $(i,j) \in \boldsymbol{P}$

$$f_{r,i}, f_{s,i}, f_{i,t} \geq 0 \qquad \text{for } i \in \boldsymbol{O}$$

$$f_{t,s}, f_{s,r} \geq 0$$

where,

$$c_{i,j} = d_i - (d_j - p_j) \ \text{ for } (i,j) \in \boldsymbol{P}$$

$$c_{r,j} = r_i - (d_i - p_i) \ \text{ for } i \in \boldsymbol{O}$$

$$c_{s,i} = c_{i,t} = 0 \ \text{ for } i \in \boldsymbol{O}$$

$$c_{i,z} = d_i \ \text{ for } i \in \boldsymbol{O}^E$$

$$c_{s,r} = c_{t,s} = c_{z,s} = 0$$

Similar network flow representations have been proposed for the optimal timing problem (Wennink (1995), and Brandimarte and Maiocco (1999).)

In problem (MCNF), the dual variable π_i for the mass balance constraint of a node *i* is called *the potential* for this node. One should note that the optimal dual solution to problem (MCNF) is not necessarily unique. However by setting $\pi_s = \pi_r = \pi_t = 0$ we ensure that π^* is also an optimal solution to problem (P). With this specific choice of node potentials we assure that π_i is the same as the lateness of operation *i* in the primal problem (P), i.e. $\pi_i = t_i + p_i - d_i = y_i - x_i$. Further if π^* is an optimal solution to problem (P) it also defines the starting time of operations, and hence the schedule.

When we are given an optimal dual solution π to problem (MCNF) we can derive so-called schedule optimal π^* from it by simply adding $-\pi_s$ to π. This step is usually necessary when we get the dual variables from a network solver code.

We can also find the optimal flow f^* with respect to a given set of optimal node potentials π^*. This requires computing the reduced cost $c^{\pi}_{i,j}$ for every arc in the MCNF problem. Then, using the reduced costs in conjunction with the complementary slackness conditions, we can transform the MCNF problem into a maximum flow problem in which the optimal solution is f^* (see Ahuja et al. (1993) pp. 316 for details.)

Given an optimal flow f^* for an instance of problem (MCNF), we can also construct a set of optimal node potentials π^* by solving a longest path problem in the residual graph $G^R(f^*)$. In this longest path problem we use arc costs in problem (MCNF), $c_{i,j}$, as arc lengths and then find the longest path from source node *s* to all other nodes. The concept of relating a schedule to a longest path problem provides significant insight relative to the literature on classical (makespan) job shop problems, which rely heavily on a similar longest path problem. In classical job shop problems, one needs to solve a longest path problem from a source node to a sink node to find starting times and the makespan. Although the topology of the our problem and that of the classical job shop problem are not exactly the same, the similarity motivates us to investigate the possibility of adapting some of the theoretical work on the classical job shop problem to our more general case.

4. An Adjacent Pairwise Interchange Search Neighborhood

Local search using *adjacent pairwise interchange* neighborhood structures have proven successful on many scheduling problems. In particular, Nowicki and Smutnicki (1996) develop a very effective local search algorithm for classical (makespan) job shop scheduling. The key to the success of their approach is a set of easily computable criteria that can rule out many neighboring solutions (interchanges) as non-improving. In most scheduling problems checking the feasibility of a neighboring solution and finding the new schedule corresponding to a selected neighbor typically require significant amount of computation. Nowicki and Smutnicki (1996) produce excellent results by greatly reducing the computation necessary to search a neighborhood. In this section we develop similar neighborhoods for our much more general problem, and show that Nowicki and Smutnicki's neighborhood is a special case of ours.

Let σ be a complete operation sequence that we want to build an adjacent pairwise neighborhood upon. Further let P_σ be a set of feasible precedence relationships and S_σ be the minimal machine precedence relationship set in P_σ. We can formally define the adjacent pairwise interchange neighborhood in the solution space as follows:

$$N_S^0(\sigma) = \left\{\sigma_{i,j} \mid (i,j) \in S_\sigma\right\}$$

where $\sigma_{i,j}$ is an operation sequence derived from σ by swapping operations i and j. Let $\sigma(k,l)$ denote the operation at position l in the processing sequence on machine k with respect to an operation sequence σ. We formally define $\sigma_{i,j}$ as follows:

$$\sigma_{i,j}(k,l) = \begin{cases} j & \text{, if } \sigma(k,l) = i \\ i & \text{, if } \sigma(k,l) = j \\ \sigma(k,l) & \text{, otherwise} \end{cases} \quad \text{for } k = 1,\cdots,m \text{ and } l = 1,\cdots,n$$

Move Feasibility

For the classical job shop scheduling problem it is well-known that a feasible schedule exists only if the set of precedence relationships forms an acyclic digraph. In the following lemma we extend this result to the problem (GSP) using our MCNF problem representation.

Lemma 1 *A precedence relationship set* P *is feasible for problem (P) if and only if the corresponding precedence digraph* $G(O,P)$ *is acyclic.*

Proof. *We observe that* $G(O,P)$ *is a subgraph of* $G(V,A \cup P)$. *Further since all arcs in* $G(O,P)$ *have infinite capacity it is also a subgraph of the residual graph* $G^R(f)$ *for any flow f in* $G(V,A \cup P)$. *Suppose that digraph* $G(O,P)$ *is cyclic and there exists a cycle* $C = (n_1,n_2,\cdots,n_k,n_1)$. *The length of cycle* C *is*

$$L(C) \quad = c_{n_1,n_2} + \cdots + c_{n_{k-2},n_{k-1}} + c_{n_{k-1},n_k} + c_{n_k,n_1}$$

$$= d_{n_1} - (d_{n_2} - p_{n_2}) + \cdots + d_{n_{k-1}} - (d_{n_k} - p_{n_k}) + d_{n_k} - (d_{n_1} - p_{n_1})$$
$$= \sum_{i=1}^{k} p_{n_i}$$

since we assume positive processing times $L(C) = \sum_{i=1}^{k} p_{n_i} > 0$. *This implies that there exists a positive cycle with infinite capacity in* $G^R(f)$. *Therefore the maximum cost flow problem defined over* $G(V, A \cup P)$ *is unbounded. However this contradicts the feasibility of problem (P) since unboundedness implies its infeasibility. Thus* $G(O,P)$ *is acyclic.*□

In neighborhood $N_S^0(\sigma)$ it is possible that a move (i,j) may lead to a cycle in the precedence relationships. One method to detect cycles can be devised by deriving a topological order for the current precedence relationship graph. However the computational complexity of an efficient implementation of the topological order finding algorithm is $O(m \cdot (n-1))$ (see Ahuja et al. (1993) pp. 79 for details). Rather than employing a computationally expensive method to ensure the feasibility of moves, we propose a neighborhood of adjacent pairwise interchanges that are guaranteed to produce feasible sequences. Such a neighborhood structure is desirable since we can explore a series of feasible solutions without addressing feasibility issues explicitly.

In the classical job shop problem, the longest path problem defines the makespan. A longest path is called a *critical path,* and arcs on the critical path are called *critical arcs*. It has been shown that reversing a *critical arc* in a feasible schedule does not lead to infeasibility (Van Laarhoven et al., 1992). In GSP the longest path problem appears in the residual graph of a solution to the maximum cost network flow problem. Recall that P is the set of all precedence relationships including both job and machine precedence relationships and $G(V, A \cup P)$ is the corresponding maximum cost flow network. We observe that $G(O,P)$ is a subgraph of $G(V, A \cup P)$. Further since all arcs in $G(O,P)$ have infinite capacity it is also a subgraph of the residual graph for any flow f in $G(V, A \cup P)$. In the following theorem we show that reversal of an arc $(i,j) \in P$ does not lead to a cycle if it is a critical arc i.e. if it is on a longest path in a residual graph.

Theorem 2 *Suppose that arc* $(i,j) \in P$ *is on a longest path in residual graph* $G^R(f^*)$ *with respect to a finite optimal flow* f^* *in an acyclic digraph* $G(O,P)$. *After reversing arc* (i,j), *the resulting digraph* $G(O, P \cup (j,i)/(i,j))$ *remains acyclic.*

Proof. *Suppose that* $G(O, P \cup (j,i)/(i,j))$ *is cyclic. Then there must exist a path* P *in* $G(O,P)$ *from node i to node j which does not include arc* (i,j), *i.e.* $P = (i, n_1, n_1, \cdots, n_k, j)$. *The length of path* P *is:*

$$\begin{aligned}L(P) &= c_{i,n_1} + c_{n_1,n_2} + \cdots + c_{n_{k-1},n_k} + c_{n_k,n_j} \\ &= d_i - (d_{n_1} - p_{n_1}) + d_{n_1} - (d_{n_2} - p_{n_2}) + \cdots + d_{n_{k-1}} - (d_{n_k} - p_{n_k}) + d_{n_k} - (d_j - p_j) \\ &= d_i - (d_j - p_j) + \sum_{l=1}^{k} p_{n_l} = c_{i,j} + \sum_{l=1}^{k} p_{n_l}\end{aligned}$$

Because of positive processing time $L(P) > c_{i,j}$. However this is a contradiction since we assumed that arc (i,j) is on a longest path, thus there must be no such path P from node i to node j with a length longer than $c_{i,j}$.□

Corollary 3 *Suppose that there exists an arc $(i,j) \in \boldsymbol{P}$ with $f^*_{i,j} > 0$ for a finite optimal flow f^* in an acyclic digraph $G(O,P)$. After reversing arc (i,j), the resulting digraph $G(O, P \cup (j,i)/(i,j))$ remains acyclic.*

Proof. *The complementary slackness optimality conditions implies that $c^{\pi}_{i,j} = 0$ for $f^*_{i,j} > 0$. Since the reduced cost is defined as $c^{\pi}_{i,j} = c_{i,j} + \pi_i - \pi_j$, when $c^{\pi}_{i,j} = 0$ it follows that $\pi_j = \pi_i + c_{i,j}$. Consequently the arc (i,j) is on a longest path from source node s to node j. Thus Theorem 2 applies in this case.*□

Based on Corollary 3 we define a restricted version of neighborhood $N^0_S(\sigma)$ as follows:

$$N^1_S(\sigma) = \left\{\sigma_{i,j} \mid (i,j) \in S_\sigma \text{ and } f^*_{i,j} > 0 \right\}$$

Both Werner and Winkler (1995), and Brandimarte and Maiocco (1999) give similar results for the adjacent pairwise interchange search in their scheduling problems and define neighborhoods that are equivalent to $N^1_S(\sigma)$. In the next section we give compact neighborhoods by further restricting this neighborhood definition with additional conditions.

Move Quality

In the literature there are two closely related techniques for accelerating the neighborhood evaluation phase. The first technique finds an easily computable lower bound rather than the more time consuming exact solution. For example, Dell'Amico and Trubian (1993) and Taillard (1994) use direct lower bounding techniques in solving the classical job shop problem with tabu search. Similar lower bound related techniques have been proposed for other machine scheduling problems by Wennink (1995), and Brandimarte and Maiocco (1999). The second technique restricts the neighborhood definition by directly eliminating non-improving moves, thus reducing the size of neighborhood. In this case the lower bound is used directly in developing the neighborhood definition instead of using it in the evaluation phase. This neighborhood reduction technique has been successfully applied to the classical job shop problem by Matsuo et al. (1988), Van Laarhoven et al. (1992), and Nowicki and Smutnicki (1996) using the critical path interpretation of the problem.

In our GSP problem we can construct several feasible solutions for the maximum cost flow problem that would result from a possible move. Since these feasible solutions provide a lower bound on the objective function of the resulting schedule, we can use them to judge the quality of a move. In fact we use this lower bounding technique to derive a simple apriori rule for identifying some deteriorating moves. We can integrate this rule into a search neighborhood definition to further reduce the neighborhood size and hence speed up the overall search. The following theorem provides a rule for eliminating obvious deteriorating neighbors in the generalized scheduling problems.

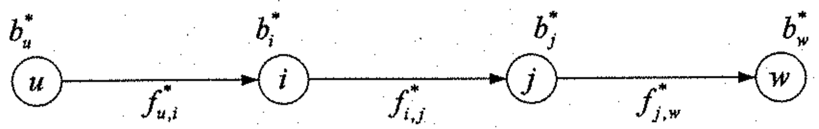

Figure 1: Subgraph G^2

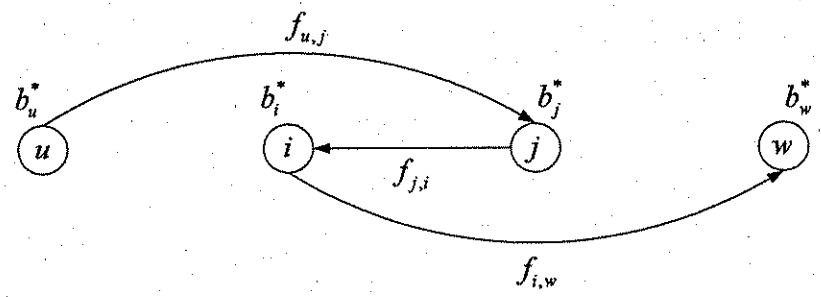

Figure 2: Subgraph G^3

Theorem 4 *Suppose that σ is a feasible operation sequence and f^* is an optimal solution to the corresponding maximum cost network flow problem. For a machine precedence arc (i,j) let operation $mp(i)$ be the immediate predecessor of operation i and operation $ms(j)$ be the immediate successor of operation j on a machine in sequence σ. If any one of the following conditions holds, then $\sigma_{i,j}$ is a non-improving neighbor:*

1. $p_j \cdot f_{mp(i),i}^* + p_i \cdot f_{j,ms(j)}^* \geq (p_i + p_j) \cdot f_{i,j}^*$
2. $(r_j + p_j - r_i) \cdot f_{r,i}^* + p_i \cdot f_{j,ms(j)}^* \geq (p_i + p_j) \cdot f_{i,j}^*$
3. $p_j \cdot f_{mp(i),i}^* + p_i \cdot f_{j,z}^* \geq (p_i + p_j) \cdot f_{i,j}^*$
4. $(r_j + p_j - r_i) \cdot f_{r,i}^* + p_i \cdot f_{j,z}^* \geq (p_i + p_j) \cdot f_{i,j}^*$

Proof. *Let $P = \{(u,i),(i,j),(j,w)\}$ be a path in $G(V, A \cup P_\sigma)$. We can partition $G(V, A \cup P_\sigma)$ into two subgraphs:*

$$G^1 = G\left(V \setminus \{u,i,j,w\}, (A \cup P_\sigma) \setminus \{(u,i),(i,j),(j,w)\}\right)$$

$$G^2 = G\left(\{u,i,j,w\}, \{(u,i),(i,j),(j,w)\}\right)$$

The subgraph G^2 is illustrated in Figure 1. In this partitioning we add pseudo supply/demand values at each node in order to isolate G^2 in the maximum cost network problem. The reversal of precedence (i,j) in P results in subgraph,

$$G^3 = G\left(\{u,i,j,w\},\{(u,i),(j,i),(j,w)\}\right)$$

which is illustrated in Figure 2. Any feasible solution for G^3 *will be a feasible solution for the new overall maximum cost flow problem after the reversal of arc* (i,j) *provided that the following conditions are satisfied:*

- *Flows in* G^1 *remain the same.*
- *The mass balance constraints are still satisfied at node* u, i, j *and* w.
- *The constructed new flows are feasible, i.e.* $f_{i,w}, f_{u,j}, f_{j,i} \geq 0$.

Since we maintain all other flows, we can define the following pseudo node supply/demand values:

$$b_u^* = -f_{u,i}^* = -f_{u,j}$$

$$b_i^* = f_{u,i}^* - f_{i,j}^* = f_{j,i} - f_{i,w}$$

$$b_j^* = f_{i,j}^* - f_{j,w}^* = f_{u,j} - f_{j,i}$$

$$b_w^* = f_{j,w}^* = f_{i,w}$$

From the above equation set we get the following solution for the constructed flows

$$f_{i,w} = f_{j,w}^* \tag{6}$$

$$f_{u,j} = f_{u,i}^* \tag{7}$$

$$f_{j,i} = f_{u,i}^* - f_{i,j}^* + f_{j,w}^*$$

Since f^* *is an optimal solution* $f_{j,w}^*$ *and* $f_{u,i}^*$ *are already nonnegative,* $f_{i,w}$ *and* $f_{u,j}$ *are also nonnegative flows by Equations 6 and 7. The new schedule associated with* $\sigma_{i,j}$ *is non-improving if* $\sigma_{i,j}$ *is a feasible sequence and* $Z_P^*(\sigma_{i,j}) \geq Z_P^*(\sigma) = Z_D^*(\sigma)$. *The weak duality theorem implies that* $Z_P^*(\sigma_{i,j}) \geq Z_D(\sigma)$. *Therefore* $\sigma_{i,j}$ *is a non-improving neighbor if the following condition holds:*

$$Z_D(\sigma) - Z_D^*(\sigma_{i,j}) \geq 0 \tag{8}$$

The above objective function difference depends on the flows in G^2 *and* G^3 *as the flows in* G^1 *remain the same. By substituting back the constructed flows given above:*

$$\begin{aligned} Z_D(\sigma) - Z_D^*(\sigma_{i,j}) &= c_{u,j}\cdot f_{u,j} + c_{j,i}\cdot f_{j,i} + c_{i,w}\cdot f_{i,w} - (c_{u,i}\cdot f_{u,i}^* + c_{i,j}\cdot f_{i,j}^* + c_{j,w}\cdot f_{j,w}^*) \\ &= c_{u,j}\cdot f_{u,i}^* + c_{j,i}\cdot (f_{u,i}^* - f_{i,j}^* + f_{j,w}^*) + c_{i,w}\cdot f_{j,w}^* - (c_{u,i}\cdot f_{u,i}^* + c_{i,j}\cdot f_{i,j}^* + c_{j,w}\cdot f_{j,w}^*) \\ &= (c_{u,j} + c_{j,i} - c_{u,i})\cdot f_{u,i}^* - (c_{j,i} + c_{i,j})\cdot f_{i,j}^* + (c_{j,i} + c_{i,w} - c_{j,w})\cdot f_{j,w}^* \end{aligned} \tag{9}$$

By Equation 9 the condition stated in Inequality 8 becomes

$$(c_{u,j} + c_{j,i} - c_{u,i})\cdot f_{u,i}^* + (c_{j,i} + c_{i,w} - c_{j,w})\cdot f_{j,w}^* \geq (c_{j,i} + c_{i,j})\cdot f_{i,j}^*$$

Since $c_{j,i} + c_{i,j} = p_i + p_j$ *we can write:*

$$(c_{u,j}+c_{j,i}-c_{u,i})\cdot f^*_{u,i}+(c_{j,i}+c_{i,w}-c_{j,w})\cdot f^*_{j,w}\geq(p_i+p_j)\cdot f^*_{i,j} \tag{10}$$

Clearly the right hand side of Inequality 10 is always nonnegative. In the following we investigate four different paths in $G(V, A\cup P_\sigma)$ *based on the structure of the maximum cost flow network.*

- ***Case 1:*** $u = mp(i)$ *and* $w = ms(j)$

  In this case path P *consists of arcs representing intermediate precedence relationships on a machine. If operation* i *is the first operation on a machine then operation* u *does not exist and we assume* $f^*_{mp(i),i}=0$. *Similarly if operation* j *is the last operation then we set* $f^*_{j,ms(j)}=0$. *Now the flow coefficients in Inequality 10 are*

$$c_{mp(i),j}+c_{j,i}-c_{mp(i),i} = \quad d_u-(d_j-p_j)+d_j-(d_i-p_i)-d_u+(d_i-p_i)=p_j$$

$$c_{j,i}+c_{i,ms(j)}-c_{j,ms(j)} = \quad d_j-(d_i-p_i)+d_i-(d_w-p_w)-d_j+(d_w-p_w)=p_i$$

  By substituting back the above flow coefficient in Inequality 10 and rearranging we get

$$p_j\cdot f^*_{mp(i),i}+p_i\cdot f^*_{j,ms(j)}\geq(p_i+p_j)\cdot f^*_{i,j} \tag{11}$$

  Because of nonnegative processing times assumption and nonnegativity of optimal flows the following is a valid inequality:

$$(p_i+p_j)\cdot(f^*_{mp(i),i}+f^*_{j,ms(j)})\geq p_j\cdot f^*_{mp(i),i}+p_i\cdot f^*_{j,ms(j)} \tag{12}$$

  By combining Inequalities 11 and 12 we get

$$(p_i+p_j)\cdot(f^*_{mp(i),i}+f^*_{j,ms(j)})\geq(p_i+p_j)\cdot f^*_{i,j}$$

  Or simply,

$$f^*_{mp(i),i}-f^*_{i,j}+f^*_{j,ms(j)}=f_{j,i}\geq 0$$

  Thus the constructed flow is feasible if Inequality 11 holds.

- ***Case 2:*** $u = r$ *and* $w = ms(j)$

  In this case node u *is the release node* r *and node* w *is the successor of operation* j *on the same machine. Now the flow coefficients are*

$$c_{r,j}+c_{j,i}-c_{r,i} = \quad r_j-(d_j-p_j)+d_j-(d_i-p_i)-r_i+(d_i-p_i)=r_j+p_j-r_i$$

$$c_{j,i}+c_{i,ms(j)}-c_{j,ms(j)} = \quad d_j-(d_i-p_i)+d_i-(d_{ms(j)}-p_{ms(j)})-d_j+(d_{ms(j)}-p_{ms(j)})=p_i$$

  By substituting the above flow coefficients Inequality 10 becomes:

$$(r_j+p_j-r_i)\cdot f^*_{r,i}+p_i\cdot f^*_{j,ms(j)}\geq(p_i+p_j)\cdot f^*_{i,j} \tag{13}$$

  In order to show feasibility of flow $f_{j,i}$ *we need to consider the following two possible values of* $f^*_{r,i}$:

  a) $f^*_{r,i}=0$

  In this case the constructed flow is $f_{j,i}=-f^*_{i,j}+f^*_{j,ms(j)}$. *Now Inequality 13 reduces to:*

$$p_i \cdot f^*_{j,ms(j)} \geq (p_i + p_j) \cdot f^*_{i,j} \tag{14}$$

Since $(p_i + p_j) \cdot f^*_{j,ms(j)} \geq p_i \cdot f^*_{j,ms(j)}$ *we can replace the left hand side of Inequality 14:*

$$(p_i + p_j) \cdot f^*_{j,ms(j)} \geq (p_i + p_j) \cdot f^*_{i,j}$$

The above inequality implies that $f^*_{j,ms(j)} \geq f^*_{i,j}$. *Therefore* $-f^*_{i,j} + f^*_{j,ms(j)} = f_{j,i} \geq 0$.

b) $f^*_{r,i} > 0$

By complementary slackness optimality conditions, $f^*_{r,i} > 0$ *implies that* $t_i = r_i$. *Similarly* $f^*_{i,j} > 0$ *implies that* $t_i + p_i = r_i + p_i = t_j$. *Since* f^* *is the feasible solution* $r_i + p_i = t_j \geq r_j$, *or simply* $r_j - r_i \leq p_i$. *By adding* p_j *to both side of the previous inequality we get* $r_j + p_j - r_i \leq p_i + p_j$ *and using this inequality we can write:*

$$\begin{aligned}(r_j + p_j - r_i) \cdot f^*_{r,i} + p_i \cdot f^*_{j,ms(j)} &\leq (p_i + p_j) \cdot f^*_{r,i} + p_i \cdot f^*_{j,ms(j)} \\ &\leq (p_i + p_j) \cdot (f^*_{r,i} + f^*_{j,ms(j)})\end{aligned} \tag{15}$$

By combining Inequalities 13 and 15 we get

$$(p_i + p_j) \cdot (f^*_{r,i} + f^*_{j,ms(j)}) \geq (p_i + p_j) \cdot f^*_{i,j}$$

Or simply,

$$f^*_{r,i} + f^*_{j,ms(j)} \geq f^*_{i,j}$$

Therefore,

$$f^*_{r,i} - f^*_{i,j} + f^*_{j,ms(j)} = f_{j,i} \geq 0$$

- ***Case 3:*** $u = mp(i)$ *and* $w = z$

In this case we assume that operation j *is an end-operation and node* w *is the terminal node* z. *If operation* j *is not an end-operation we set* $f^*_{j,z} = 0$. *Now the flow coefficients are*

$$\begin{aligned} c_{mp(i),j} + c_{j,i} - c_{mp(i),i} &= \quad d_{mp(i)} - (d_j - p_j) + d_j - (d_i - p_i) - d_{mp(i)} + (d_i - p_i) = p_j \\ c_{j,i} + c_{i,z} - c_{j,z} &= \quad d_j - (d_i - p_i) + d_i - (d_w - p_w) - d_j + (d_w - p_w) = p_i \end{aligned}$$

Inequality 10 becomes:

$$p_j \cdot f^*_{mp(i),i} + p_i \cdot f^*_{j,ms(j)} \geq (p_i + p_j) \cdot f^*_{i,j}$$

We observe that this case is same as Case 1 and the argument given for the feasibility of flow $f_{j,i}$ *is valid.*

- ***Case 4:*** $u = r$ *and* $w = z$

In this case node u *is the release node* r *and operation* j *is an end operation so that node* w *is the terminal node* z. *The flow coefficients in Inequality 10 become*

$$c_{r,j} + c_{j,i} - c_{r,i} = r_j - (d_j - p_j) + d_j - (d_i - p_i) - r_i + (d_i - p_i) = r_j + p_j - r_i$$

$c_{j,i} + c_{i,z} - c_{j,z} = \ d_j - (d_i - p_i) + d_i - d_j = p_i$

Inequality 10 becomes:

$$(r_j + p_j - r_i) \cdot f^*_{r,i} + p_i \cdot f^*_{j,z} \geq (p_i + p_j) \cdot f^*_{i,j}$$

This case is the same as Case 2 and the analysis given for showing the feasibility of flow $f_{j,i}$ *applies.*□

The conditions given in Theorem 4 are sufficient but not necessary conditions to identify a non-improving neighbor. However, these conditions can be evaluated much faster than solving the maximum cost flow problem corresponding to a move. The following corollary shows that the feasibility restriction based on critical machine arcs only eliminates some of the non-improving neighbors without eliminating any of the improving moves.

Corollary 5 *Suppose that* σ *is a feasible sequence and* f^* *is an optimal solution to the corresponding maximum cost network flow problem. If* $f^*_{i,j} = 0$ *for a machine precedence arc* (i, j), *then* $\sigma_{i,j}$ *is a non-improving neighbor.*

Proof. *When* $f^*_{i,j} = 0$ *the first condition in Theorem 4 becomes:*

$$p_j \cdot f^*_{mp(i),i} + p_i \cdot f^*_{j,ms(j)} \geq (p_i + p_j) \cdot f^*_{i,j} = 0$$

Since we assume positive processing times and $f^* \geq 0$, *this condition is always satisfied and* (i, j) *is a non-improving move.*□

Compact Adjacent Pairwise Interchange Neighborhoods

Using Corollary 3 and Theorem 4 we can define the following two compact neighborhoods,

$$N^2_S(\sigma) = \ \left\{\sigma_{i,j} \mid (i,j) \in S_\sigma,\ f^*_{i,j} > 0 \text{ and } \Delta_{i,j} = \max\{\Delta^1_{i,j}, \Delta^2_{i,j}, \Delta^3_{i,j}, \Delta^4_{i,j}\} \leq 0\right\}$$

$$N^3_S(\sigma) = \ \left\{\sigma_{i,j} \mid (i,j) \in S_\sigma,\ f^*_{i,j} > 0 \text{ and } \Delta_{i,j} = \max\{\Delta^1_{i,j}, \Delta^2_{i,j}, \Delta^3_{i,j}, \Delta^4_{i,j}\} < 0\right\}$$

where,

$$\Delta^1_{i,j} = \quad p_j \cdot f^*_{mp(i),i} + p_i \cdot f^*_{j,ms(j)} - (p_i + p_j) \cdot f^*_{i,j}$$

$$\Delta^2_{i,j} = \quad (r_j + p_j - r_i) \cdot f^*_{r,i} + p_i \cdot f^*_{j,ms(j)} - (p_i + p_j) \cdot f^*_{i,j}$$

$$\Delta^3_{i,j} = \quad p_j \cdot f^*_{mp(i),i} + p_i \cdot f^*_{j,z} - (p_i + p_j) \cdot f^*_{i,j}$$

$$\Delta^4_{i,j} = \quad (r_j + p_j - r_i) \cdot f^*_{r,i} + p_i \cdot f^*_{j,z} - (p_i + p_j) \cdot f^*_{i,j}$$

Here note that $Z^*\left(\sigma_{i,j}\right) \geq Z^*(\sigma) + \Delta_{i,j}$ by Theorem 4. The only difference between $N^2_S(\sigma)$ and $N^3_S(\sigma)$ is that we also exclude potentially neutral neighbors in $N^3_S(\sigma)$ by imposing strict inequality. It is usually favorable to exclude neutral moves in a neighborhood since they may cause the neighborhood search to cycle. However including neutral moves in a neighborhood definition can sometimes be beneficial as long as appropriate cycle prevention mechanisms are in place (c.f. Crauewels et al. (1998).) Obviously, $N^3_S(\sigma) \subseteq N^2_S(\sigma) \subseteq N^1_S(\sigma) \subseteq N^0_S(\sigma)$.

A Special Case: Classical Job Shop Problem

We incorporate the flow information provided by the maximum cost network flow problem in the definitions of $N_S^2(\sigma)$ and $N_S^3(\sigma)$. However one does not necessarily need to solve a maximum cost network flow problem to use this information for all scheduling problems. For regular scheduling problems it is usually a simple matter to generate such information (dual solutions) since there are no deliberate idle times in a schedule. In this section we consider the classical job shop problem where arc flows are either 0 or 1, and they can be easily determined by examining the longest path solution.

In the case of the classical job shop problem, the solution to the maximum cost network flow is a critical path from node r to node z containing arcs with $f_{i,j}^* = 1$. On this critical path a *block* is defined as a maximal set of adjacent critical path operations that are to be processed on the same machine. Further an operation is called a *border operation* if it is the first or last operation of a block, otherwise it is an *internal operation*. Van Laarhoven et al. (1992) present a neighborhood that is based on swapping adjacent operations of a block. Clearly, $N_S^1(\sigma)$ is the same neighborhood for the classical job shop problem since an adjacent operation pair (i, j) of a block has a flow $f_{i,j}^* = 1$ by definition, i.e. arc (i, j) is a critical machine precedence arc. Nowicki and Smutnicki (1996) give a compact neighborhood, referred as *NS neighborhood*, by eliminating some of the neighbors from Van Laarhoven et al.'s neighborhood. In particular they only consider the operation pairs including one border operation, i.e. the first or the last precedence machine relationship of a block. They also exclude the first operation pair in the first block and the last operation pair in the last block in their neighborhood.

Corollary 6 *In the classical job shop scheduling problem the* $N_S^3(\sigma)$ *neighborhood is the same as NS neighborhood.*

Proof. *Using the block notion given above we evaluate all possible adjacent pairwise interchanges in a job shop schedule in terms of Theorem 4:*

1. Both operations i and j are internal operations:

In this situation $f_{mp(i),i}^* = f_{j,ms(j)}^* = f_{i,j}^* = 1$ *and the first condition becomes* $p_j + p_i - (p_i + p_j) = 0$.

2. One of operations i and j is an internal and the other is a border operation:

Suppose that operation i *is the internal and operation* j *is the border operation. Now* $f_{mp(i),i}^* = f_{i,j}^* = 1$ *and* $f_{j,ms(j)}^* = 0$ *and the first condition in Theorem 4 is* $p_j - (p_i + p_j) = -p_i < 0$.

3. Operation i is the first operation of the first block:

In this case the second condition applies with $r_i = r_j = 0$ *and* $f_{r,i}^* = f_{i,j}^* = f_{j,ms(j)}^* = 1$. *This condition evaluates as* $p_j + p_i - (p_i + p_j) = 0$.

4. Operation j is the last operation of the last block:

With $f_{mp(i),i}^* = f_{i,j}^* = f_{j,z}^* = 1$ *the third condition becomes* $p_j + p_i - (p_i + p_j) = 0$.□

Corollary 6 shows that $N_S^3(\sigma)$ is the same as NS neighborhood for the classical job shop problem. This is an important result since the NS neighborhood is the most compact neighborhood known. A tabu search algorithm based on this neighborhood has provided some of the best known solutions to well-known benchmark problems, and is considered one of the most effective search neighborhoods for this problem (Vaessens, 1996, Jain and Meeran, 1999 and Jain et al., 2000).

Since the lower bounding technique presented in Theorem 4 is quite general and readily applicable to a wide variety of both regular and non-regular scheduling problems it means that we can produce compact adjacent pairwise interchange neighborhoods for these scheduling problems as well. For example we already applied this neighborhood definition in several advanced solution methodologies and achieved excellent results on both our own E/T test problems and some total weighted tardiness job shop problems from the literature (Avci, 2001.) Theorem 4 also has an immediate consequence for the classical job shop scheduling problem: we are able to introduce release times into the adjacent pairwise interchange neighborhood definition, which is essential for dealing with dynamic classical job shop problems, but not available in NS neighborhood definition.

Recently Brandimarte and Maiocco (1999) underlined the need for extending the block properties to non-regular scheduling problem since they have been most useful for solving the classical job shop scheduling problem. We believe that our network flow analysis and related lower bounding scheme for the adjacent pairwise interchange neighborhood provides a key step in linking regular and non-regular scheduling problems in terms of both theory and application. Further the relationship between the network flow presentation and the block properties may help in extending branch and bound methods based on the block properties to non-regular scheduling problems. This is a particularly interesting research topic since developing a branch and bound method for a non-regular scheduling problem has been considered quite challenging due to the idle time insertion requirements (Hoogeven and Van de Velde, 1996).

Connectivity and Optimality

A search neighborhood N defines a digraph in the solution space. In our problem the solution space is the set of feasible sequences Ω. Therefore, the nodes of the neighborhood digraph are a feasible sequence in Ω and an arc (σ, τ) indicates that sequence τ is a neighbor of sequence σ under neighborhood *N*, i.e. $\tau \in \Omega(\sigma)$. The following *connectivity property* is defined by using the concept of a *neighborhood digraph*:

Definition 1 [Vaessens, 1995] *A neighborhood function* N *is called* strongly connected *if the corresponding digraph is strongly connected. A neighborhood function* N *is called* optimum connected *if for each solution there exists a path to an optimal solution in the corresponding neighborhood digraph.*

Van Laarhoven et al. (1992) prove that neighborhood $N_S^1(\sigma)$ is optimum connected for the classical job shop problem. They use this property to show that the standard simulated annealing algorithm asymptotically converges to an optimal solution under certain conditions. Wennink (1995) adapts their proof technique to show that the connectivity property holds for the adjacent pairwise interchange neighborhood in his generalized scheduling problem. Similarly, it is trivial to show that neighborhood $N_S^1(\sigma)$ has the connectivity property in the generalized scheduling problem by again following Van Laarhoven et al.'s proof.

Nowicki and Smutnicki (1996) report that the connectivity does not hold for neighborhood $N_S^3(\sigma)$ in the classical job shop scheduling problem. We have shown by a counter example that neighborhoods $N_S^2(\sigma)$ and $N_S^3(\sigma)$ also do not have the connectivity property for the generalized scheduling problem (see Avci, 2001).

For the classical job shop scheduling problem Nowicki and Smutnicki give a sufficient optimality condition where a feasible schedule is also an optimal schedule if its $N_S^3(\sigma)$ neighborhood is empty. The counter example in Avci (2001) proves that emptiness of neighborhoods $N_S^2(\sigma)$ and $N_S^3(\sigma)$ does not imply optimality in our generalized scheduling problem. These negative results regarding the connectivity property and sufficient optimality condition for $N_S^2(\sigma)$ and $N_S^3(\sigma)$ seem to have little impact in practice.

5. Basic Local Search Algorithms

In the previous section we defined a compact adjacent pairwise interchange neighborhood for the generalized scheduling problem. In this section we address the issue of searching in this neighborhood. In particular we consider simple descent algorithms which produce a local optimal solution with respect to a given search neighborhood. We call these algorithms *basic local search* since they do not consider any ascent or hill-climbing moves to escape local optima. Basic local search algorithms are usually used for exploiting a solution found by some other solution method. For instance a solution produced by a dispatching heuristic can be improved upon by applying a basic local search algorithm to it. Although basic local search algorithms lack local minima escape capability, they are usually fast enough to be integrated in other solution methods. In fact we extensively used basic local search algorithms to enhance other solution methods which are usually good at exploring the solution space but not powerful enough in the exploitation aspect (Avci 2001.) Here we should point out that due to space limitation we choose basic local search algorithms to present computational results on $N_S^3(\sigma)$. In Avci (2001) we used our compact neighborhood in developing more sophisticated solution methods including a tabu search algorithm and achieved excellent results. We also do not include these results here due to space limitations.

A basic local search method starts with a given feasible solution and then selects an improving move from the neighborhood of this solution under a given neighborhood definition. It visits a series of solutions with decreasing objective function values and terminates when there are no further improving moves in the current neighborhood.

```
Input:    An operation sequence $\sigma$ and a neighborhood function $N$
Output:   A local optimal operation sequence $\sigma$ w.r.t. neighborhood function $N$
Step 1.   Generate neighborhood $N(\sigma)$ for operation sequence $\sigma$
Step 2.   Select a neighbor $\sigma_{i,j}$ such that $Z^*(\sigma_{i,j}) < Z^*(\sigma)$
Step 3.   If Step 2 finds an improving neighbor $\sigma_{i,j}$, set $\sigma \leftarrow \sigma_{i,j}$ and go to Step 1, otherwise stop.
```

Figure 3: Basic Local Search Algorithm

Figure 3 presents a pseudocode for a basic local search algorithm which considers strictly improving moves and avoids neutral moves and related cycling issues. In Step 2 of this algorithm we select an improving neighbor from the neighborhood of the current solution. Since there might be more than one improving neighbor one should decide how to perform this selection procedure, which is often referred as a *search strategy*.

A very simple and intuitive search strategy is to evaluate all neighbors and pick the one which yields the largest decrease in the objective function, (the so-called *best-improving search*.) This search strategy is similar to the well-known steepest descent search in optimization theory. However, when the evaluation of all neighbors is relatively expensive, best-improving search may become prohibitive to include in other solution methods. In this case, instead of evaluating all of the neighbors in a neighborhood it is often favorable to quickly identify an improving neighbor and continue with the search without any consideration of the magnitude of improvement in the objective function. This type of search strategy is often called the *first-improving search*. In this case it is important to evaluate neighbors by following a priority order which is likely to give a higher priority to improving neighbors so that an improving neighbor can be identified early in the selection process. In this study we consider four different priority functions, $\Pi_{i,j}$, to determine the priority of a move (i,j):

- Maximum cost: $\Pi_{i,j} = -c_{i,j} \cdot f_{i,j}$
- Maximum flow: $\Pi_{i,j} = -f_{i,j}$
- Minimum lower bound: $\Pi_{i,j} = \Delta_{i,j}$
- Random

Obviously the order in which we evaluate the neighbors is not relevant in a best-improving search since we have to examine all neighbors. However, in a first-improving search the priority order determines the final solution quality and the overall computational time requirement as it determines the search path in the solution space and the number of solutions visited in this path. In general the best-improving search is likely to visit a smaller number of solutions compared to first-improving search, however it may require more computational time as it evaluates all neighbors that it encounters in the course of the search.

6. Computational Results

In this section we present extensive computational results on solution space neighborhoods. All computational experiments have been performed on a IBM PC 300GL personal computer with 733 Mhz Pentium III processor and 512 MB memory. Algorithms were programmed in C++ and compiled with the Microsoft Visual C++ 6.0 compiler. We use ILOG CPLEX Callable Library 7.0 to solve optimization problems. In particular we use the ILOG CPLEX Network Optimizer module with its default settings to solve maximum cost network flow problems.

In the local search algorithms we must solve a series of very similar MCNF problems. In fact when we evaluate an adjacent pairwise interchange neighborhood of a current solution, the problems are different in at most three arcs. Although the topologies of these MCNF problems are not exactly the same, we can still use the optimal basis of the current solution, and start the CPLEX Network Optimizer from an advanced basis, to evaluate the neighboring solution. In our problem advanced basis start turns out to be quite advantageous. We performed several tests to quantify the gains from using advanced basis starts in local search. The results of these tests are reported in detail in Avci (2001). The bottom line is that we were able to reduce CPU times by about 75%.

In our computational experiments we consider the total weighted earliness and tardiness (TWET) problems in a job shop setting. We randomly generated a set of 10×10 problem instances of the static TWET job shop problems in a systematic way which vary certain factors so as to produce several different problem types. We adopted a 3-letter labeling system to identify our benchmark instances:

- The first letter indicates the routing type; random routing (R) or cluster routing (C). In a random routing each job visits every machine in a random order. In cluster routing the machine set is partitioned into two equal clusters and each job first visits all machine in the first cluster randomly then visits all machines in the second cluster in the same fashion.
- The second letter indicates the problem type; operation-level problem in which all operations have a due date, and earliness and tardiness penalties (A), or job-level problem where only the end-operation has a due date and associated earliness and tardiness penalties (E).
- The third letter indicates the due date variance factor; small due date variance (S) or large due date variance (L). We use random dispatching to generate a schedule and apply this variance factor to the completion times of operations to further generate due dates.

We have 8 different problem classes, namely CAS, RAS, CES, RES, CAL, RAL, CEL and REL. In each problem class we generated 5 instances using different random seed numbers. Therefore we have 40 benchmark instances in total. Further details on these TWET benchmark problems are given in Avci (2001).

We test several basic local search algorithms based on adjacent pairwise interchange (API) neighborhoods including: best-improving search (B) and first-improving search. For the first improving search, we tried different schemes to prioritize the order in which neighbors (adjacent pairs, or arcs) were evaluated. These schemes were: maximum cost (F-MC), maximum flow (F-MF), minimum lower bound (F-MLB) and random (F-R) priorities.

Basic local search algorithms start with a given initial solution and terminate when they find a local optimal solution with respect to neighborhood definition. For each problem we applied the earliest due date (EDD) dispatching rule to construct a feasible machine sequence and solved the corresponding MCNF problem to generate an initial feasible schedule. In the literature, the EDD dispatching rule has frequently been used in scheduling problems with due dates. There are many other dispatching rules and other solution construction techniques. However, in our computational experiments we aim to investigate the quality of our search neighborhood rather than initial solutions. For this purpose EDD schedules are a reasonable choice. In Avci (2001) we also consider several other solutions and total weighted tardiness job shop problems from the literature as well, and provide detailed computational results which are consistent with the results we include in this section.

In Table 1 we give various neighborhood related statistics (averaged over all local search algorithms) by problem class. In this table M denotes the average number of moves, or iterations, performed by local search algorithms. $\Sigma| N_S^1|$ indicates the average total number of neighbors N_S^1 encountered in the course of local search, therefore $\Sigma| N_S^1| /M$ is the average number of neighbors in a single N_S^1 neighborhood. Similarly ΣN_S^3 and $\Sigma| N_S^3| /M$ give the same information for neighborhood N_S^3. Note that all our test problems are 10×10, therefore $| N_S^0| = 90$. $R_{1,3}$ and $R_{0,3}$ denote the relative percent reduction in the average neighborhood size between neighborhoods N_S^1 and N_S^3 and neighborhoods N_S^0 and N_S^3, respectively. For example $R_{1,3} =$ 28.87 for problem class CAS means that neighborhood definition N_S^3 results in a 28.87% reduction in the neighborhood size with respect to neighborhood N_S^1. Similarly $R_{0,3} = 63.56$ means neighborhood N_S^3 eliminates 63.56% of the moves in neighborhood N_S^0, which always has 90 moves.

Problem	M	$\Sigma\|N_S^1\|$	$\Sigma\|N_S^1\|/M$	$\Sigma\|N_S^3\|$	$\Sigma\|N_S^3\|/M$	$R_{1,3}$	$R_{0,3}$
CAS	46.64	2167.96	46.23	1526.4	32.80	28.87	63.56
RAS	49.16	2051.68	41.63	1503.36	30.73	25.99	65.85
CES	20.88	282.08	13.06	266.72	12.40	5.07	86.22
RES	25.76	267.84	10.29	260.12	10.01	2.75	88.88
CAL	63.32	2887.36	45.68	2006.00	32.05	29.8	64.39
RAL	53.88	2209.44	40.69	1582.08	29.6	26.94	67.11
CEL	20.28	278.96	13.45	262.72	12.73	5.23	85.86
REL	26.44	275.04	10.41	270.20	10.24	1.6	88.63
Average	38.30	1302.55	27.68	959.70	21.32	15.78	76.31

Table 1: Summary of the API Neighborhood Statistics by Problem Type

The results in Table 1 show difference neighborhood reductions for the end-operation penalty problems (denoted _E_) and the all-operation penalty problems (denoted _A_). The average number of moves, M, clearly indicates that the problems with only end-operation penalties require significantly fewer iterations. Similarly the average number of moves in a neighborhood is also smaller in the end-operation penalty problems compared to the all-operation penalty problems. For the all-operation penalty problems, the MCNF problem is likely to contain more critical arcs (arcs with positive flow) compared to the end-operation penalty problems. This is simply because of the structure of the MCNF problem; when the number of penalized operations increases there are more flows in the network. Therefore the size of neighborhood N_S^1 is larger for these problems because neighborhood N_S^1 is defined over critical arcs, which are plentiful in these problem classes. When we apply the additional elimination conditions given in N_S^3, significant reduction results since the large number of critical arcs provides greater opportunity for these conditions to apply. On the other hand the neighborhood size for N_S^1 is quite small for the end-operation penalty problems since there are considerably fewer critical arcs in the MCNF problem. Consequently the N_S^3 conditions have a relatively smaller chance of eliminating moves in neighborhood N_S^1. For this problem type neighborhood N_S^1 provides most of the elimination from N_S^0 and neighborhood N_S^3 yields a relatively smaller further reduction. Table 1 also shows the relationship between the search neighborhood size and the number of iteration for a basic local search. Local search algorithms perform more iterations as the size of the search neighborhood gets larger.

Algorithm	M	N	$\Sigma\|N_S^1\|$	$\Sigma\|N_S^1\|/M$	$\Sigma\|N_S^3\|$	$\Sigma\|N_S^3\|/M$	$R_{1,3}$	$R_{0,3}$
B	31.50	797.38	1067.35	27.77	797.38	21.55	15.31	76.06
F-MC	48.73	451.43	1690.45	27.02	1202.68	20.34	17.11	77.40
F-MF	35.48	257.95	1168.55	27.61	882.43	21.48	15.26	76.14
F-MLB	32.63	244.95	1099.03	28.10	831.30	21.96	14.81	75.60
F-R	43.15	171.63	1487.35	27.90	1084.73	21.27	16.41	76.37
Average	38.30	384.67	1302.55	27.68	959.70	21.32	15.78	76.31

Table 2: Summary of the API Neighborhood Statistics by Algorithm Type

In Table 2 we provide a summary of results for neighborhood characteristics, which are similar to those given in Table 1, in terms of basic local search algorithms. These results show that the average neighborhood sizes are independent of basic local search algorithms, which is an expected result. In Table 2 N is the total average number of moves (neighbors) in N_S^3 tha has been evaluated during a basic local search algorithm. We observe that the first-improving search with maximum cost priority took both the largest number of iterations and total average number of evaluated moves among the first-improving search algorithms. The random priority search resulted in the second largest number of iterations but evaluated the smallest number of moves in all search algorithms.

In the best-improving search strategy we need to evaluate all moves in all neighborhoods we encounter during the search. On the other hand the first-improving search employs a priority rule to determine an order of evaluation which may help identify improving moves early in the evaluation phase without examining all moves. In Table 3 E denotes the percentage of moves in neighborhood N_S^3 that have been evaluated and I denotes the percentage of improving moves among these evaluated moves. For instance we evaluated on the average 32.27% of the moves while conducting a first-improving search with maximum cost priority on problem class CAS. Only 9.86% of the evaluated moves were improving and resulted in a new solution. Our results indicate that the random priority ordering evaluates considerably less moves and yields more improving ones. Among the remaining priority rules the maximum cost rule considers the highest percentage of moves and finds the lowest percentage of improving moves, thus it is inferior when compared to other priority rules.

	Max. Cost		Max. Flow		Min. LB		Random	
Problem	E	I	E	I	E	I	E	I
CAS	32.27	9.86	29.08	10.53	28.23	11.62	15.97	20.00
RAS	40.37	8.55	28.36	11.36	29.23	11.54	14.64	23.08
CES	42.70	20.00	35.80	23.20	37.28	22.46	29.58	28.34
RES	35.78	30.06	33.68	30.59	30.06	34.71	26.92	36.63
CAL	37.11	9.07	29.30	10.68	26.29	11.80	11.77	27.20
RAL	39.19	9.36	26.37	12.60	32.36	10.34	15.85	22.60
CEL	44.89	19.89	38.81	21.49	34.80	23.27	23.88	31.13
REL	40.33	27.14	38.14	27.84	33.07	31.35	27.39	34.45
Average	39.08	16.74	32.44	18.54	31.42	19.64	20.75	27.93

Table 3: Effect of First-Improving Search Priority Rules

We use the following definition of the objective function gap, $\Delta(Z_A,Z_B)$, in minimization problems for an objective function value Z_A with respect to another objective function value Z_B such that $Z_A \geq Z_B$:

$$\Delta(Z_A, Z_B) = 100 \cdot \left(1 - \frac{Z_A}{Z_B}\right)$$

We found a lower bound for our TWET problems using a time-indexed IP formulation given in Avci (2001). We use this lower bound for measuring the objective function gap in the TWET problems, since they are all positive numbers, i.e. $Z_B > 0$ and we do not have the optimal solutions to these problems.

			First-Improving			
Problem	EDD	Best-Improving	Max. Cost	Max. Flow	Min. LB	Random
CAS	63.03	34.17	33.33	31.16	31.41	37.04
RAS	70.79	31.88	43.58	29.58	35.69	40.72
CES	72.00	56.99	59.01	54.59	53.87	52.09
RES	89.04	76.59	77.37	74.68	77.31	76.87
CAL	67.64	32.16	30.35	32.27	31.33	32.50
RAL	69.55	35.43	40.52	30.49	34.67	39.49
CEL	73.08	56.08	58.34	53.87	55.78	55.92
REL	89.12	77.41	80.04	75.47	76.56	76.21
Average	74.28	50.09	52.82	47.76	49.58	51.36

Table 4: Percentage Objective Function Gap Summary

In Table 4 we tabulate objective function gap information on basic local search algorithms with an EDD schedule start. In this table we also tabulate the objective function gap for the EDD initial solution. When we compare different cases in the TWET test problems we observe that problems with only end-operation penalties have significantly larger gaps compared to problems with the all-operation penalties. These results show that local search provide a significant improvement in all problem classes.

In Table 5 we give the average total CPU time in seconds for different problem types and basic local search algorithms. These results indicate that all basic local search algorithms are fast enough to be integrated in other solution methods. In Tables 1-3 we observe the relationship between neighborhood size, length of basic local search and number of moves evaluated during a search. The CPU times reported in Table 5 are consistent with our earlier observations. The CPU time is mainly determined by the number of MCNF problems solved during a search. Among search algorithms the best-improving search requires considerably more CPU time. Similarly all-operation penalty problems have larger search neighborhoods and the local search algorithms spend more CPU time in these problems.

		First-Improving			
Problem	Best-Improving	Max. Cost	Max. Flow	Min. LB	Random
CAS	0.30	0.17	0.10	0.10	0.10
RAS	0.38	0.18	0.13	0.13	0.13
CES	0.04	0.03	0.02	0.02	0.02
RES	0.06	0.03	0.02	0.02	0.02
CAL	0.37	0.28	0.13	0.13	0.13
RAL	0.38	0.19	0.11	0.11	0.11
CEL	0.06	0.03	0.03	0.03	0.03
REL	0.07	0.03	0.02	0.02	0.02
Average	0.21	0.12	0.07	0.07	0.07

Table 5: Total CPU Time (sec.) Summary for the Basic Local Search Algorithms

7. Conclusions

In this paper, we present new results on neighborhood structures for a broad class of scheduling problems. These results allow us to extend some of the most effective algorithms for the classical (makespan) job shop problem to a much more general set of problems. Included in this more general set are problems with non-regular performance measures. These problems are important in practice, but have seldom been addressed in the literature. We have demonstrated that the proposed neighborhoods have the potential to be embedded in more sophisticated local search algorithms due the speed with which they can be evaluated. The demonstration of this potential may be found in (Avci, 2001) (and hopefully a subsequent paper). Among the results in Avci (2001), we develop local search algorithms for the generalized scheduling problem which are applicable to both regular and non-regular problems. We then apply these algorithms to the well know weighted tardiness (WT) job shop problems of Pinedo and Singer (1999). We show that our algorithms perform at least on a par with those of Pinedo and Singer (if not slightly better) even though they make no attempt to exploit the regularity of the WT problems.

References

(Book) R. Ahuja, T. Magnanti, and J. Orlin, Network Flows (1 ed.). Prentice Hall, Inc., 1993.

(Ph.D thesis) S. Avci, "Algorithms for generalized non-regular scheduling problems", Department of Industrial and Manufacturing Systems Engineering, Lehigh University, 2001.

(Journal article) K. Baker and G. Scudder, "Sequencing with earliness and tardiness", Operations Research, vol. 38, pp. 22-36, 1990.

(Journal article) P. Brandimarte and M. Maiocco, "Job shop scheduling with a non-regular objective: A comparison of neighborhood structures based on sequencing/timing decomposition", International Journal of Production Research, vol. 37, pp. 1697-1715, 1999.

(Journal article) H. Crauwels, C. Potts, and L. Van Wassenhove, "Local search heuristics for the single machine total weighted tardiness scheduling problem", INFORMS Journal on Computing, vol. 10, pp. 341-350, 1998.

(Journal article) J. Davis and J. Kanet, "Single-machine scheduling with early and tardy completion costs", Naval Research Logistics, vol. 40, pp. 85-101, 1993.

(Journal article) J. Du and J. Leung, "Minimizing total tardiness on one machine is NP-hard", Mathematics of Operations Research, vol. 15, pp. 483-495, 1990.

(Journal article) M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem", Annals of Operations Research, vol. 41, pp. 231-252, 1993.

(Journal article) B. Faaland and T. Schmitt, "Scheduling tasks with due dates in a fabrication/assembly process", Operations Research, vol. 35, pp. 378-388, 1987.

(Journal article) T. Fry, R. Armstrong, and J. Blackstone, "Minimizing weighted absolute deviation in single machine scheduling", IIE Transactions, vol. 19, pp. 445-450, 1987.

(Journal article) T. Fry, R. Armstrong, K. Darby-Dowman, and P. Philpoom, "A branch and bound procedure to minimize mean absolute lateness on a single processor", Computers and Operations Research, vol. 23, pp. 171-182, 1996.

(Journal article) J. Hoogeven and S. Van de Velde, "A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time", INFORMS Journal on Computing, vol. 8, pp. 402-412, 1996.

(Journal article) A. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future", European Journal of Operational Research, vol. 113, pp. 390-434, 1999.

(Journal article) A. Jain, B. Rangaswamy, and S. Meeran, "New and ``stronger" job-shop neighbourhoods: A focus on the method of nowicki and smutnicki (1996)", Journal of Heuristics, vol. 6, pp. 457-480, 2000.

(Journal article) J. Kanet and V. Sridharan, "Scheduling with inserted idle time: Problem taxonomy and literature review", Operations Research, vol. 48, pp. 99-110, 2000.

(Technical Report) H. Matsuo, C. Suh, and R. Sullivan, "A controlled search simulated annealing method for the general jobshop scheduling problem", Graduate School of Business, University of Texas, Austin, TX., Technical Report 03-04-88, 1988.

(Journal article) E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem", Management Science, vol. 42, pp. 797-813, 1996.

(Journal article) M. Pinedo and M. Singer, "A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop", Naval Research Logistics, vol. 46, pp. 1-17, 1999.

(Conference proceedings) R. Storer, S. Wu, and R. Vaccari, "Local search in problem and heuristic space for job shop scheduling genetic algorithms", In New Directions for Operations Research in Manufacturing: Proceedings of a Joint US/German Conference, Gaithersburg, Maryland, USA, July 30-31, 1991, Berlin; New York, pp. 149-160. under the auspices of Operations Research Society of America (ORSA), Deutsche Gesellschaft für Operations Research (DGOR): Springer-Verlag, 1992.

(Journal article) E. Taillard, "Parallel taboo search techniques for the job shop scheduling problem", ORSA Journal on Computing, vol. 6, pp. 108-117, 1994.

(Ph. D. thesis) R. Vaessens, "Generalized Job Shop Scheduling: Complexity and Local Search", Department of Mathematics and Computer Science, Eindhoven University of Technology, 1995.

(Journal article) P. Van Laarhoven, E. Aarts, and J. Lenstra, "Job shop scheduling by simulated annealing", Operations Research, vol. 40, pp. 113-125, 1992.

(Ph. D. thesis) M. Wennink, "Algorithmic Support for Automated Planning Boards", Department of Mathematics and Computer Science, Eindhoven University of Technology, 1995.