SENSITIVITY ANALYSIS FOR SCHEDULING PROBLEMS

Nicholas G. Hall [*]

Marc E. Posner [‡]

[*] Max M. Fisher College of Business
The Ohio State University
hall.33@osu.edu

[‡] Industrial Engineering
The Ohio State University
posner.1@osu.edu

June 5, 2001

Abstract

This paper addresses a variety of new issues related to sensitivity analysis for scheduling problems. Although this work focuses on scheduling problems, many of our questions and solution approaches can be applied to other optimization problems. The applicability of these issues is illustrated with examples from well known scheduling models. We provide fast methods to determine when a previously optimal solution remains optimal. Other methods restore an optimal solution after a parameter change. For some problems, sensitivity analysis results depend on the positions of the jobs with changed parameters. Also, the value of studying the sensitivity of an optimal sequence instead of the sensitivity of an optimal schedule is demonstrated. We identify situations where performing additional or different computations during optimization facilitates sensitivity analysis. Selection among multiple optimal schedules is considered in order to improve the robustness of an optimal schedule. We discuss which types of sensitivity analysis questions are intractable because the scheduling problem is intractable. We also examine the use of heuristic error bounds when the data is modified.

Key Words and Phrases: manufacturing and scheduling, sensitivity analysis, optimization, heuristics.

OR/MS Index 1989 Subject Classification:

production/scheduling; sequencing, deterministic.

production/scheduling; approximations/heuristic.

Classical sensitivity analysis addresses "What if ...?" types of questions that arise from parameter changes. For example, if one or more cost coefficients change, how does this affect an optimal solution? Does the cost of the solution change, while the decisions leading to the optimal outcome remain unchanged? Is there a need to identify new decisions to restore optimality? Or, if the original solution is used when a parameter change makes it no longer optimal, how much is lost by continuing to use the original solution? In the context of scheduling problems, this paper presents a variety of new research issues related to sensitivity analysis.

Sensitivity analysis is important because real world problems exist in a changing environment (Anderson *et al.*, 1999). For example, customer demands and specifications, prices of raw materials, and employee and equipment availability, are difficult to estimate and subject to change. As a result, estimates of these values are often used (Hillier and Lieberman, 1995). Sometimes the parameters may be deliberately overstated or understated, to promote the interests of the estimator (Hillier and Lieberman). This suggests the need to identify the effects of changes on the quality of the solution.

For general integer programming problems, Roberts (1990), Cozzens and Roberts (1991), and Mahadev *et al.* (1998) use measurement theory to address robustness. A number of authors examine sensitivity analysis issues for branch and bound algorithms. Examples include Roodman (1972), Piper and Zoltners (1976), Marsten and Morin (1977), Geoffrion and Nauss (1977), and Schrage and Wolsey (1985). An alternative approach is to apply duality theory at the optimal solution, which is found by using cutting plane methods. Authors who use related approaches include Klein and Holm (1979), Wolsey (1981) and Cook *et al.* (1986).

Van Hoesel and Wagelmans (1999) derive some negative results for sensitivity analysis. They show that finding the sensitivity analysis range for an individual cost coefficient in an intractable zero-one optimization problem is not possible in polynomial time, unless $P = NP$ (see Garey and Johnson, 1979, for related definitions).

Kolen *et al.* (1994) study the class of scheduling heuristics that is known as list scheduling rules. For several heuristics, they derive bounds on the number of possible assignments of jobs to machines. For a nonlinear optimization model, Sotskov (1991) describes the *stability radius*, i.e. the maximum amount of parameter change that maintains optimality. Related works include

Kravchenko *et al.* (1995) and Sotskov *et al.* (1995). Tovey (1986) considers the problem of minimizing makespan on a changing number of identical processors. Similar studies of precedence constrained scheduling appear in Graham (1975) and Tovey (1990).

A number of authors consider sensitivity analysis where multiple parameter changes are allowed. These works include Bradley *et al.* (1977), Wendell (1985), Wondolowski (1991) and Wendell (1992). Exact solution methods are presented by Gal (1979) and Camm and Burwell (1991). Simulation approaches are considered by several authors including Wagner (1995) and James and Buchanan (1996). Chakravarti and Wagelmans (1998) and Sotskov *et al.* (1998) consider bounds for an ϵ-optimal solution to remain ϵ-optimal.

Comprehensive overviews of sensitivity analysis are provided by Gal (1979), Wagelmans (1990) and Greenberg (1998). The latter reference contains an extensive bibliography.

In Section 1 of this work, we define our notation, describe several classical sensitivity analysis questions, and present a variety of new issues related to sensitivity analysis. In the remainder of this work, we provide examples and applications of these issues. These applications show that the issues we present are useful. Section 2 presents *a priori* sensitivity results that can be applied without using information about an optimal schedule. Section 3 contains a discussion of sensitivity analysis for problems that are solvable in polynomial time. This section examines several issues, including situations where the position of the job with changing parameters affects the sensitivity analysis. We also identify conditions under which standard linear programming sensitivity analysis can be improved, both in establishing optimality ranges and in finding new optimal schedules. Section 4 presents situations where planning for sensitivity analysis makes it worthwhile to perform additional work or work of a different type, when finding an optimal schedule. We also study the selection among multiple optimal schedules to improve the robustness of an optimal sequence with respect to parameter changes. Section 5 examines intractable problems. First, we consider the computational complexity of performing sensitivity analysis. Then, examples are provided of how to derive bounds on the change in optimal cost as the result of a parameter change. We also use the information contained in an optimal schedule for the original data to obtain, for the modified data, a new heuristic schedule with a guaranteed error bound. Finally, Section 6 contains a conclusion and some possible extensions.

1 Preliminaries

In this section, we define our notation, describe several classical sensitivity analysis questions, and introduce a variety of new and important issues related to sensitivity analysis.

1.1 Notation

Let $N = \{1, \ldots, n\}$ denote the set of jobs to be processed on m machines, $M_1, \ldots, M_m$. Let r_j, p_j, w_j, d_j and $\bar{d}_j$ denote the known release date, processing time, weight, due date and deadline of job j, respectively, for $j \in N$. We assume throughout that $r_j, p_j, w_j, d_j, \bar{d}_j \geq 0$ and integer for $j \in N$. Let π denote a job *sequence*, or order. A *schedule* σ provides the start and completion time of every job or job piece. For all of the problems we consider, once π and a machine assignment are given, the unique corresponding σ can be determined in $O(n)$ time.

Sensitivity analysis considers the effects of modifying problem parameters. We use the convention that parameters and variables of the modified problem are identified with an added $'$, e.g. p_i becomes p_i'. Let $C_j(\sigma)$ denote the completion time of job j in schedule σ before parameter changes occur, and $C_j'(\sigma)$ denote the completion time of job j in schedule σ after parameter changes occur. The start time of job j in schedule σ is specified by $S_j(\sigma)$. When obvious, $C_j(\sigma)$, $C_j'(\sigma)$ and $S_j(\sigma)$ are replaced by C_j, C_j' and S_j, respectively. We let σ^* denote an optimal schedule before parameter changes occur and σ' denote an optimal schedule after parameter changes occur. Similarly, π^* and π' represent optimal job sequences before and after there are parameter changes, respectively. Let $C_j^* = C_j(\sigma^*)$ and $C_j'' = C_j'(\sigma')$. We use similar notation for start times. Let $z(\sigma)$ and $z'(\sigma)$ denote the cost of schedule σ for the original and modified problems, respectively. Also, let $z^* = z(\sigma^*)$ and $z'' = z'(\sigma')$. Let $E_j = \max\{d - C_j, 0\}$ and $T_j = \max\{C_j - d, 0\}$ denote the earliness and tardiness of job j, respectively, relative to a common due date, d. Let Δ denote the amount of a parameter change. Also, let $\overline{\Delta}$ and $\underline{\Delta}$ denote the supremum and infimum, respectively, on Δ such that, depending on the context, σ^*, π^* or z^* remains optimal.

The standard classification scheme for scheduling problems (Graham *et al.*, 1979) is $\psi_1|\psi_2|\psi_3$, where ψ_1 defines the scheduling environment, ψ_2 describes the job characteristics or restrictive requirements, and ψ_3 defines the objective function to be minimized. When $\psi_1 = Pm$, P denotes the identical parallel machine environment. If $m = 1$, then we omit P. For ψ_2, we may use

r_j:	release dates
$\bar{d}_j$:	deadlines
$p_j = 1$:	unit processing times
$d \geq \sum p_j$:	unrestrictively large common due date
$pmtn$:	job preemption.

The objectives we consider under ψ_3 are the minimization of

$C_{\max}$:	makespan
$\sum C_j$:	total completion time
$\sum w_j C_j$:	total weighted completion time
$\sum U_j$:	total number of late jobs
$\sum w_j U_j$:	total weighted number of late jobs
$L_{\max}$:	maximum lateness
$\sum (E_j + T_j)$:	total earliness and tardiness.

1.2 From Classical Questions To New Issues

We describe several classical sensitivity analysis questions that have both theoretical and practical importance. These questions apply to situations in which complete information is known about σ^*. Then, one or more data values in that instance changes in a known way. Some questions that are important to consider are:

Q1. What are the limits to a parameter change such that the solution remains optimal?

Q2. Given a specific change of a parameter, what is the new optimal cost?

Q3. Given a specific change of a parameter, what is a new optimal solution?

Q4.1–Q4.3. What differences occur in answering Q1 – Q3, respectively, when several parameters change simultaneously?

Questions Q1 – Q4 are classical sensitivity analysis questions that are often considered only in specific settings. Linear programming is one example. For this class of models, sensitivity analysis is thoroughly explored for all of the parameters (see Gal, 1979).

Typically, when a parameter change exceeds the bounds specified by Q1, a new solution is found by parametric analysis. This approach finds new optimal solutions and values as the parameter changes, until the desired change is reached. However, a manager may want an answer for a specific change of a parameter, suggesting a more direct approach. An answer to Q2 is a solution to the *evaluation version* of a problem. Whereas, an answer to Q3 is a solution to the *optimization version*. A solution to Q3 implies a solution to Q2. However, the converse is not generally true (Papadimitriou and Steiglitz, 1982).

Regarding Q3, the way in which a schedule changes may be of interest. For many production operations, it is important to maintain a schedule that does not change too much, a situation with low "schedule nervousness" (Steele, 1973, Federgruen and Tzur, 1994). For any problem, it is possible to answer Q4.1 – Q4.3 by repeating the analysis for each parameter change. However, we are interested in more efficient approaches. Also, there may be special combinations of parameter changes that are worthwhile to consider.

The main purpose of this work is to identify and discuss new issues related to sensitivity analysis. Several classical scheduling problems are used to illustrate these issues, and to show that answers to these issues are obtainable.

Sensitivity Analysis Issues

I1. When do sensitivity analysis approaches apply to general classes of problems?

I2. How can sensitivity analysis for several parameter changes be performed more efficiently than by repeating the computational work needed for an individual change?

I3. When is sensitivity analysis more appropriate for π^* than for σ^*?

I4. When is sensitivity analysis more appropriate for z^* than for σ^*?

I5. What types of sensitivity analysis do not require the full details of the solution?

I6. How does the position in σ^* of the job with changing parameters affect sensitivity analysis?

I7. What portions of the optimal schedule remain the same after parameter changes?

I8. To facilitate sensitivity analysis, when is it advantageous to find the optimal solution by performing extra or different types of computational work?

I9. When is it useful to evaluate the robustness of optimal solutions?

I10. What is the computational complexity of answering sensitivity analysis questions for intractable scheduling problems?

I11. What types of heuristic sensitivity analysis can be developed for NP-hard problems?

We now provide some motivation for issues I1 through I11. Additional motivation is presented throughout the paper. Issue I1 investigates when a sensitivity analysis approach can be applied to an entire class of problems. Such results aid in analyzing other problems in that class.

For issue I2, we study situations where multiple parameter changes are independent and others where the changes are dependent. In many real world problems, multiple parameter changes are related. This may be due to the relationships between the parameters, such as a correlation between the value and work required. It may also be due to the modelling process. An example is a problem with discounted costs. When the base cost of a job changes, this induces a set of cost changes, one for each time period.

Issue I3 considers situations where it is useful to consider sensitivity analysis for a sequence. Frequently, a production process is controlled by the job sequence and not the actual times provided by a schedule. Thus, the need to reschedule occurs only when π^* changes. Also, even a very small change in a processing time changes an optimal schedule. On the other hand, larger and more meaningful ranges on parameter changes may be obtained when considering π^*.

Issue I4 considers situations where it is useful to consider sensitivity analysis for the optimal cost. This issue examines the *set* of optimal solutions. Changes are considered for which the original optimal set no longer contains an optimal solution with cost z^*. One application where this issue is important is when we are allocating additional resources to reduce processing times. In this situation, we are interested in the minimum reduction to the processing time of a job that provides a less expensive schedule.

If issue I5 can be resolved without knowledge of an optimal solution, then it is possible to estimate the effects of changes in resources or requirements in advance of detailed planning. Issue I5 is also important where some information about σ^* is missing or unreliable.

For issue I6, the analysis may depend on either the original position in σ^*, the new position, or both. The positions may affect the change in optimal cost, the quality of a bound or the computational effort. The computational effort may grow in proportion to a change in job position or may be polynomially bounded for some job positions while intractable for others.

In practical situations, answering issue I7 can save time and computational effort when the existing information about the original optimal schedule is used. In addition, users of the schedule can continue to implement the parts of the schedule that do not change.

Issue I8 is valuable for planning many types of sensitivity analysis in advance, in order to improve the overall efficiency and usefulness of the results. Sometimes it is beneficial to store information gathered during the optimization phase. In other cases, the use of sensitivity analysis suggests the use of an entirely different and more complicated optimization procedure. Finally, the type of sensitivity to be performed may affect the choice of solution procedure.

The robustness of a solution is its insensitivity to parameter change. This is important in many applications and is considered in issue I9. For a problem with multiple optimal solutions, it may be useful to choose one which is most robust to a particular parameter change. Negative results that show the impossibility of making such a choice are also valuable.

For issue I10, understanding the computational complexity of sensitivity analysis directs the design of appropriate types of sensitivity analysis methods. Moreover, the large number of computational complexity results available for scheduling problems can be used to imply related results for sensitivity analysis of those problems.

Such intractability results for sensitivity analysis problems motivate the use of heuristic sensitivity analysis approaches. Regarding Issue I11, one useful type of analysis is to derive heuristic bounds on the quality of solutions obtained by simple sensitivity analysis procedures. It is also interesting to show when the bounds to answer Q1 are only sufficient and when they are exact.

The issues in our list are applicable to various classical questions. Issues I1, I5, I6, I8, I10 and I11 address all of the classical questions. Issue I2 addresses Q4.1 – Q4.3. Issue I3 addresses Q1, Q3, Q4.1 and Q4.3. Issue I4 addresses Q1, Q2, Q4.1 and Q4.2. Issue I7 addresses Q3 and Q4.3. Issue I9 addresses Q1 and Q4.1. In the following subsection, we indicate which of these potential applications of issues to the classical questions are addressed in our paper.

1.3 Overview of Results

Table 1 summarizes, for our results, the connections between the classical sensitivity analysis questions and the newly defined issues. Because of space limitations, we address only a subset of example applications. However, we provide at least one application for each issue.

Where multiple results appear within the same row, they illustrate different aspects of an issue. Regarding issue I2, Theorem 4 considers two dependent parameter changes for a list scheduling problem, and Theorem 8 considers multiple independent changes for a problem that

	Q1	Q2	Q3	Q4.1	Q4.2	Q4.3
I1				Thm 2	Thm 2	Thm 2
I2				Thm 4,10		Thm 8,9
I3	R10, Thm 6,13,14		Thm 7	Thm 4		
I4	Thm 12					
I5	R1–R10			Thm 1		
I6		Thm 3,16		Thm 1	Thm 5	Thm 5
I7						Thm 5
I8	Thm 12				Thm 11	Thm 11
I9	Thm 13,14					
I10		Thm 15	Thm 15			
I11	Thm 17,18	Thm 16,19,20	Thm 20			

Table 1: Results Connecting Questions and Issues.

requires solution by linear programming. Theorems 9 and 10 consider a single parameter change that implies other changes because of a concise formulation. Theorem 9 is concerned with reoptimization and Theorem 10 is concerned with finding optimality ranges. For issue I3, result R10 considers a situation in which very limited information is available about the optimal solution. In Theorem 4, only information about adjacent jobs is needed. Theorem 6 demonstrates how studying an optimal sequence instead of an optimal schedule can provide necessary and sufficient conditions for continued optimality. Theorem 7 shows how efficient reoptimization is possible using the sequence. Finally, Theorems 13 and 14 provide positive and negative results, respectively, about the possibility of identifying the most robust sequence. For issue I5, results R1–R10 use no knowledge about the solution. Theorem 1 uses only knowledge about the last job on each machine. For issue I6, Theorem 3 demonstrates the effect of job position on the cost for a single parameter change. Whereas, Theorems 1 and 5 consider the schedule when there are multiple parameter changes. A particular job position is easy to analyze in Theorem 1, and the effect of job position in heuristic analysis is considered in Theorem 16. For issue I8, different types of planning for sensitivity analysis are illustrated. Theorem 11 shows how to save time in sensitivity analysis by storing additional information during optimization. Whereas, Theorem 12 suggests an entirely new optimization procedure that facilitates sensitivity analysis. For issue I9, Theorem 13 shows when and how a robust schedule can be identified, while Theorem 14 identifies situations where this is not possible. For issue I11, Theorem 16 finds a heuristic bound on new optimal cost for a given parameter change. Theorems 17 and 18 find exact limits on parameter change for continued optimality in heuristic sensitivity analysis. Finally, Theorem 19 provides heuristic bounds on the cost of the current optimal solution, while Theorem 20 provides heuristic bounds on the cost of a modified solution.

2 A Priori Results

Motivated by issue I5, we examine what can be said about Q1 – Q3 *before* a specific optimal schedule is found. Geoffrion and Nauss (1977) provide some answers to this question. They discuss sensitivity analysis results based on elementary observations for general integer programs. The basic results they present, stated using scheduling terminology, are:

GN1. If the set of feasible schedules is reduced and σ^* is still feasible, then σ^* is still optimal.

GN2. In a minimization problem, if job k completes at its lower bound in σ^* and the cost of k increases, then σ^* is still optimal.

GN3. In a minimization problem, if job k completes at its upper bound in σ^* and the cost of k decreases, then σ^* is still optimal.

We adapt these results to scheduling problems. For most scheduling problems with regular measures (Rinnooy Kan, 1976), the following parameter changes do not alter an optimal schedule.

From GN1:

R1: An increase in a release date, but not later than the time when the job starts processing.

R2: A decrease in a deadline, but not earlier than the time when the job completes processing.

R3: The addition of a precedence constraint requiring job i to be processed before job k when $C_i^* \leq S_k^*$.

From GN2:

R4: An increase in the weight of a job that is processed first.

R5: An increase in the weight of a job that starts processing at its release date.

From GN3:

R6: A decrease in the weight of a job that completes processing at its deadline.

R7: A decrease in the weight of a job that is processed last in a single machine problem.

It is possible to derive additional results, similar to those above, that do not follow from GN1 – GN3. Result R8 describes a sufficient condition under which an increase in the set of feasible schedules does not change the optimal schedule in nonpreemptive problems that have regular measures and do not have precedence constraints.

R8: An increase in $\bar{d}_k$ when there is consecutive idle time of at least $\max_{j \in N}\{p_j \mid C_j^* > C_k^*\}$ in the interval $[C_k^*, \bar{d}_k]$.

A bottleneck cost problem is one where the objective is to minimize the largest single cost incurred in the schedule. An example is minimizing the maximum lateness.

R9: In bottleneck objective problems, an increase in processing time that does not create a new bottleneck or increase the current bottleneck.

Some extensions of these results are possible. An example is a multiple machine version of R7. The following result addresses issues I5 and I6 by answering Q4.1. This result is an example where sensitivity analysis is polynomially bounded for specific job positions and type of analysis. Contrast this result with the intractability of Q1 for changes in job weights, as proved by Van Hoesel and Wagelmans (1999).

Theorem 1 *For identical parallel machine problems with $\sum w_j C_j$ objective, if the weight of the last job on each machine is reduced by the same constant Δ, then σ^* remains optimal.*

Proof. Reindex the jobs so that i is the last job to complete on machine M_i in σ^* for $i = 1, \ldots, m$. Suppose we reduce the weight of jobs $1, \ldots, m$ by Δ and σ^* is no longer optimal. Let σ' be optimal, and let h_i be the last job to finish on M_i in σ' for $i = 1, \ldots, m$. Now, $\sum_{i=1}^m C''_{h_i} = \sum_{j=1}^n p_j$. From the optimality of σ', $|C''_{h_i} - C''_{h_q}| \leq \min\{p_{h_i}, p_{h_q}\}$ for $1 \leq i \neq q \leq m$. Thus, $\sum_{i=1}^m C''_i \leq \sum_{i=1}^m C''_{h_i}$. This implies that $\sum_{i=1}^m \Delta C''_i \leq \Delta \sum_{j=1}^n p_j$. Since σ^* is no longer optimal,

$$z'(\sigma') < z'(\sigma^*) = z(\sigma^*) - \Delta \sum_{j=1}^n p_j.$$

If we increase the weight of jobs $1, \ldots, m$ by Δ, then

$$z(\sigma') = z'(\sigma') + \Delta \sum_{i=1}^m C''_i \leq z'(\sigma') + \Delta \sum_{j=1}^n p_j < z(\sigma^*).$$

This contradicts the optimality of σ^*. $\square$

The following result describes a parameter change that does not alter π^* (issue I3).

R10: An increase of Δ in p_k, when there is idle time throughout the interval $[C_k^*, C_k^* + \Delta]$.

3 Problems that are Solvable in Polynomial Time

We first consider the class of polynomially solvable scheduling problems that are solvable by a list scheduling algorithm. We address issue I1 for all problems in this class. Then, we refine our analysis for a problem that has a nonregular performance measure, addressing issue I6. Next, an example of issue I3 is discussed. This example also addresses I2 when there are dependent parameter changes. The same scheduling problem is used to address issues I6 and I7. Next, we consider polynomially solvable scheduling problems that require an algorithm more complicated than list scheduling, and present examples for issue I3. Finally, we examine problems that can be solved in polynomial time by variants of linear programming. For this class of problems, we consider issue I2 for questions Q4.1 and Q4.3. We also provide an improvement on the optimality range found by classical sensitivity analysis and its general purpose extensions.

3.1 Problems Admitting Optimal List Schedules

The simplest category of problems that we consider are those that are optimally solvable by a *list scheduling rule*. Such problems are called list scheduling problems. For these problems, a priority list is determined from pairwise comparisons of the characteristics of each individual job. Then, jobs are selected for processing in list sequence. Some scheduling problems that are solvable by a list scheduling rule include $1||\sum w_jC_j$ (Smith, 1956), $P||\sum C_j$ (Conway *et al.*, 1967), $1||L_{\max}$ (Jackson, 1955) and $1|d \geq \sum p_j|\sum(E_j + T_j)$ (Kanet, 1981).

To analyze the time complexity required to answer sensitivity analysis questions for list scheduling problems, we introduce some notation. Let $\mathcal{S}$ be the computational time required to compare two jobs in a list. In the problems we consider, $O(\mathcal{S})$ is a constant. Given a list, let $\mathcal{T}$ be the time required to determine both a schedule, which includes all job start or completion times, and its cost. Throughout this work, $\mathcal{T} = O(n)$. Let μ be the number of simultaneous parameter changes that are considered. The first result addresses issue I1 by establishing the time complexity of answering Q4.1 – Q4.3 for all list scheduling problems.

Theorem 2 *For list scheduling problems, if a set of parameter changes does not alter the list sequence, then the continued optimality of σ^* can be verified in $O(\mu\mathcal{S})$ time. Otherwise, a new optimal schedule and its cost can be determined in $O(\mu\mathcal{S}\log n + \mathcal{T})$ time.*

Proof. We can determine whether the position of a job changes by making comparisons between adjacent jobs in the list in $O(\mathcal{S})$ time. Consequently, the optimality of σ^* can be verified in $O(\mu\mathcal{S})$ time. When a schedule is no longer optimal, it is necessary to resequence some jobs. The new optimal position for each such job can be found by pairwise comparisons with other jobs in $O(\mathcal{S}\log n)$ time. Since there are at most μ such jobs, this requires $O(\mu\mathcal{S}\log n)$ time. A new optimal schedule can then be found in $O(\mathcal{T})$ time. $\square$

Although Theorem 2 answers Q4.1 – Q4.3 for list scheduling problems, there are other issues to address. One issue is when does sensitivity analysis depend on the position of the job with changing parameters. For $1|d \geq \sum p_j|\sum(E_j + T_j)$, we show how to answer Q2 in $O(\log n)$ time. This is less than the $O(\mathcal{S} + \mathcal{T}) = O(n)$ time given by Theorem 2. Algorithm KANET (Kanet, 1981), finds an optimal schedule for this problem.

Algorithm KANET

1. Reindex the jobs such that $p_1 \geq \cdots \geq p_n$.
2. Job $2i - 1$ is placed in position i, for $i = 1, \ldots, \lceil n/2 \rceil$.
   Job $2i$ is placed in position $n - i + 1$, for $i = 1, \ldots, \lfloor n/2 \rfloor$.
3. Schedule job $\lceil n/2 \rceil$ to complete at d. Schedule all other jobs without inserted idle time.

We first describe a procedure that finds a new optimal job sequence when there is a single parameter change of the form $p'_k = p_k + \Delta$, where $\Delta > 0$. The case where $\Delta < 0$ is similar.

Algorithm ETCHG

1. Find the position of p'_k in the nonincreasing processing time sequence. Let i denote this position, where $i < k$.
2. Jobs $1, \ldots, i - 1$, and jobs $k + 1, \ldots, n$ retain their positions in σ'.
3. Job k occupies in σ' the position that job i occupies in σ^*.
4. Jobs from $i, \ldots, k - 1$ that preceded d in σ^* follow d in σ', and vice versa. These jobs are sequenced by increasing indices before d and decreasing indices after d.

We need the following preliminary result.

Lemma 1 *For some $k \in N$, let $p'_k = p_k + \Delta$ be a parameter change for $1|d \geq \sum p_j| \sum(E_j + T_j)$, where $\Delta > 0$. Then, Algorithm ETCHG finds a new optimal job sequence.*

Proof. By construction. $\square$

Let C^*_{-1} denote the start time of job 1. The next result addresses issue I6.

Theorem 3 *For some $k \in N$, let $p'_k = p_k + \Delta$ be a parameter change for $1|d \geq \sum p_j| \sum(E_j + T_j)$, where $\Delta > 0$. Also, let i be the new position of job k in the nonincreasing processing time sequence. The cost of a new optimal schedule is*

$$z'' = z^* + \lfloor i/2 \rfloor \Delta - \lceil (k-i)/2 \rceil p_k + \begin{cases} C^*_{k-2} - C^*_{i-2}, & \text{if } k \text{ and } i \text{ are both odd} \\ C^*_{k-1} - C^*_{i-2}, & \text{if } k \text{ is even and } i \text{ is odd} \\ C^*_{k-1} - C^*_{i-1}, & \text{if } k \text{ and } i \text{ are both even} \\ C^*_{k} - C^*_{i-1}, & \text{if } k \text{ is odd and } i \text{ is even.} \end{cases}$$

Further, this new optimal cost can be found in $O(\log n)$ time.

Proof. From Lemma 1, ETCHG finds a new optimal schedule. In each of the four cases in the statement of the theorem, the costs of jobs $k+1, \ldots, n$ do not change. As part of this analysis, observe that $p_k = C^*_k - C^*_{k-2}$ when $k \geq 1$ is odd, and $p_k = C^*_k - C^*_{k+2}$ when $k \geq 2$ is even.

Case 1. k and i are odd, i.e. $C^*_i, C^*_k < d$.

The total change in cost for jobs $1, 3, \ldots, i-2$ is

$$\lfloor \frac{i}{2} \rfloor \Big[(C^*_{i+1} - C^*_{k+1}) - (C^*_{k-2} - C^*_{i-2}) + \Delta \Big].$$

The total change in cost for jobs $2, 4, \ldots, i-1$ is

$$\lfloor \frac{i}{2} \rfloor \Big[(C^*_{k-2} - C^*_{i-2}) - (C^*_{i+1} - C^*_{k+1}) \Big].$$

The total change in cost for jobs $i, i+2, \ldots, k-2$ is

$$\frac{(k-i)}{2} \Big[(C^*_{k+1} - d) - (d - C^*_{k-2}) \Big] + (C^*_{k-2} - C^*_{i-2}).$$

The total change in cost for jobs $i+1, i+3, \ldots, k-1$ is

$$\frac{(k-i)}{2} \Big[(d - C^*_{k}) - (C^*_{k+1} - d) \Big] - (C^*_{i+1} - C^*_{k+1}).$$

The total change in cost for job k is $C^*_{i+1} - C^*_{k+1}$. Adding the above terms establishes Case 1 of the theorem. The proofs of the other three cases are similar.

Although ETCHG constructs a new optimal schedule, the cost of that schedule can be computed from z^* without implementing ETCHG. Finding i, the new position of job k, requires $O(\log n)$ time. Given i, the optimal cost can be computed in constant time. Thus, the overall time requirement is $O(\log n)$. $\square$

While Theorem 2 gives general guidelines, additional analysis may be needed to provide efficient results for specific problems. Such an example is $1\,||\sum w_jC_j$. Suppose we simultaneously change parameters p_k and w_k for a given job $k \in N$. In a real world example, the value of a job may be based on the raw material cost plus a term for the added labor. Thus,

$$w_k = \tau p_k + K, \tag{1}$$

where K is the raw material cost and τ is the added value per unit of processing time. If p_k changes, then we expect w_k to change also. When the value of the weight is given by (1), the new value of w_k is $w'_k = w_k + \tau\Delta$ if $p'_k = p_k + \Delta$.

Assume that jobs are indexed such that $w_1/p_1 \geq \cdots \geq w_n/p_n$. We now provide an example of issues I2 and I3 that answers Q4.1.

Theorem 4 *For some $k \in N$, let $p'_k = p_k + \Delta$ and $w'_k = w_k + \tau\Delta$ be parameter changes for $1||\sum w_jC_j$. Then, π^* remains optimal if and only if Δ is within the following ranges.*

i. *If $\tau < w_{k+1}/p_{k+1}$, then*

$$(w_kp_{k-1} - w_{k-1}p_k)/(w_{k-1} - \tau p_{k-1}) \leq \Delta \leq (w_kp_{k+1} - w_{k+1}p_k)/(w_{k+1} - \tau p_{k+1}).$$

ii. *If $w_{k+1}/p_{k+1} \leq \tau \leq w_{k-1}/p_{k-1}$, then*

$$\max\{(w_{k+1}p_k - w_kp_{k+1})/(\tau p_{k+1} - w_{k+1}), (w_kp_{k-1} - w_{k-1}p_k)/(w_{k-1} - \tau p_{k-1})\} \leq \Delta < \infty.$$

iii. *If $\tau > w_{k-1}/p_{k-1}$, then*

$$(w_{k+1}p_k - w_kp_{k+1})/(\tau p_{k+1} - w_{k+1}) \leq \Delta \leq (w_{k-1}p_k - w_kp_{k-1})/(\tau p_{k-1} - w_{k-1}).$$

Furthermore, the above conditions can be checked in constant time.

Proof. The stated inequalities follow from the conditions that keep the list order unchanged. $\square$

Theorem 4 provides bounds on modifications to the parameters of job k such that the optimal sequence does not change. When the sequence does change, it is interesting to consider how much the schedule is disrupted (issue I7), and how much this disruption affects the computational work required (issue I6). The next result examines the extent of schedule disruption that results from a parameter change and provides specific answers to Q4.2 and Q4.3.

Theorem 5 *For some $k \in N$, let $p'_k = p_k + \Delta$ and $w'_k = w_k + \tau\Delta$ be parameter changes for $1||\sum w_j C_j$. If $w_{j-1}/p_{j-1} \geq w'_k/p'_k \geq w_j/p_j$, then the times at which the jobs $1, \ldots, \min\{k-1, j-1\}$ are processed do not change. Further, jobs $\max\{k+1, j\}, \ldots, n$ are processed in the same sequence. Also, if $j < k$, then jobs $j, \ldots, k-1$ are processed in the same sequence, and if $j > k$, then jobs $k+1, \ldots, j-1$ are processed in the same sequence. The time required to compute a new optimal schedule and its cost is proportional to $|k-j|$.*

Proof. The new job position and sequence follow from the fact that a WSPT sequence is optimal.

The cost of a new optimal schedule is

$$z^* + \begin{cases} p_k \sum_{i=j}^{k-1} w_i - w_k \sum_{i=j}^{k-1} p_i, & j < k \\ -p_k \sum_{i=k+1}^{j-1} w_i + w_k \sum_{i=k+1}^{j-1} p_i, & j > k. \end{cases} \quad \square$$

Theorem 5 is an example of position dependent sensitivity analysis. The time required to compute the new optimal schedule and cost depends on the change in position of job k between the original and the modified list sequence.

3.2 Other Problems that are Solvable in Polynomial Time

The analysis in Theorem 2 demonstrates that sensitivity analysis for list scheduling problems can be performed quickly, and sometimes without using all of the information about σ^*. However, for problems that require more complicated but still polynomial time procedures, the full details of σ^* may be essential for efficient sensitivity analysis. This answers issue I5 in the negative. A problem of this type is $1|r_j, pmtn|\sum C_j$. The shortest remaining processing time (SRPT) rule proposed by Miller and Schrage (1966) finds an optimal schedule for this problem in $O(n \log n)$ time. For this problem, we first examine when π^* changes, issue I3. By performing sensitivity analysis for π^*, we answer Q1 in Theorem 6.

The following result characterizes an optimal schedule for problem $1\,|r_j, pmtn|\sum C_j$.

Lemma 2 *(Schrage, 1968) An optimal schedule for $1|r_j, pmtn|\sum C_j$ can be found by processing the available job with the shortest remaining processing time at each point in time. Further, this schedule contains no more than $n-1$ preemptions.*

The next result identifies a useful relationship between π^* and σ^*. While π^* does not completely characterize a schedule in preemptive problems, it does so for $1\,|r_j, pmtn|\sum C_j$ when σ^* is found using the SRPT rule.

Lemma 3 *An optimal schedule for $1|r_j, pmtn|\sum C_j$ is uniquely characterized by a job sequence.*
Proof. From the SRPT rule, a job can be preempted by job j only at time r_j. Hence, each period of processing that does not complete a job ends at the release date of the subsequent job in the sequence. Thus, the sequence of jobs processed completely characterizes a schedule. $\square$

When the release date for a job changes, the next preliminary result establishes time intervals within which the optimal schedule does not change.

Lemma 4 *For some $k \in N$, let $r'_k = r_k + \Delta$ be a parameter change for $1|r_j, pmtn|\sum C_j$. Suppose that $\pi' = \pi^*$. If job k preempts job i in σ^*, then σ' and σ^* are identical outside the time interval $[\min\{r_k, r'_k\}, C^*_i]$.*
Proof. Because the selections by the SRPT rule are unchanged in the interval $[0\,, r_k]$, σ' and σ^* are identical before the release date of job k. Since $\pi' = \pi^*$ by assumption, only jobs that are processed in the interval $[\min\{r_k, r'_k\}, C^*_i]$ for π^* are processed in this interval for π'. Further, the SRPT rule ensures that if job j is processed in $[\min\{r_k, r'_k\}, C^*_i]$, then j starts and completes in that interval. Thus, after C^*_i, σ' and σ^* are identical. $\square$

Lemma 4 is an example of a position dependent result for a scheduling problem that both is solvable in polynomial time and does not have a known optimal list scheduling rule. Lemma 4 illustrates issues I6 and I7. Unlike Theorem 5, this result depends only on the location of the job in the original optimal schedule and not on its new location.

We assume that the jobs are indexed such that $r_1 \leq \cdots \leq r_n$. Moreover, if $r_j = r_{j+1}$, then $p_j \leq p_{j+1}$. To break ties when using the SRPT rule, we assume that the job with the smallest index is selected for processing. Define σ^* and σ' to be optimal SRPT schedules, where $r'_k = r_k + \Delta$. Also, job i is the last job in process before S^*_k. Let

$\hat{S}_j(t)$	$=$	start time of the last period of processing of job j before time t in σ^*
$R_j(t)$	$=$	remaining processing time of job j at time t in σ^*
$\alpha_j(t)$	$=$	size of the first piece of job j that starts no earlier than time t in σ^*, if there are multiple pieces
$I(t_0, t_1)$	$=$	total idle time in the interval $[t_0, t_1]$ in σ^*
$\mathcal{E}$	$=$	$\{j \mid$ job j starts earlier in σ' than in σ^* for some $\Delta < 0\}$
e	$=$	first job to begin processing in σ^* after all jobs in $\mathcal{E}$, and before C_i^* when $\mathcal{E} \neq \emptyset$
$\mathcal{L}$	$=$	$\{j \mid$ job j starts later in σ' than in σ^* for some $\Delta > 0\}$.

We first show how to calculate $\mathcal{E}$ and $\mathcal{L}$, given $\pi' = \pi^*$ and a parameter change of the form $r_k' = r_k + \Delta$, for some $k \in N$, for $1|r_j, pmtn|\sum C_j$.

Procedure FIND$\mathcal{E}$: for $\Delta < 0$

0. Let job i be the last job in process before job k starts. Set $\mathcal{E} = \emptyset$.
1. If $C_i^* = S_k^*$, then terminate.
2. Set $\mathcal{E} = \{k\}$. Let $j = k$.
3. Let q be the job such that $C_j^* = S_q^*$. If $r_q = S_q^*$ or if q does not exist, then terminate.
4. Set $\mathcal{E} = \mathcal{E} \cup \{q\}$, and $j = q$. Go to Step 3.

Procedure FIND$\mathcal{L}$: for $\Delta > 0$

1. Set $\mathcal{L} = \{k\}$. Let $j = k$.
2. Let $q = \min\{l \in N \mid C_j^* \leq S_l^*$ and no processing occurs in the interval $[C_j^*, S_l^*]\}$. If q does not exist, then terminate.
3. Set $\mathcal{L} = \mathcal{L} \cup \{q\}$, and $j = q$. Go to Step 2.

Next, we present two lemmas that identify necessary and sufficient conditions for π^* to remain optimal after a release date change.

Lemma 5 *For some $k \in N$, let $r_k' = r_k + \Delta$ be a parameter change for $1|r_j, pmtn|\sum C_j$. Suppose $\Delta < 0$ and suppose i is the last job that is processed before r_k in σ^*. Then, π^* is an optimal sequence for the modified problem if and only if*

a. $\Delta \geq \min\{t < S_i^* \mid I(t, S_i^*) = 0\} - r_k$, *when* $S_k^* \geq C_i^*$

b. $\Delta \geq \max\{t \mid R_j(t) \geq p_k$ *for some* j *where* $C_j^* \leq S_k^*\} - r_k$, *when* $S_k^* \geq C_i^*$

c. $\Delta > \max_{q\in\mathcal{E}} \max_{u\in N}\{p_u - R_q(r_u) \mid S^*_q < S^*_u < C^*_q,$ *no* $j \in \mathcal{E}$ *exists such that* $C^*_j \leq S^*_q$ *and* $S^*_j - r_j < R_q(r_u) - p_u\}$

d. $\Delta \geq \max_{q\in\mathcal{E}\cup\{e\}\setminus\{k\}}\{r_q - S^*_q \mid S^*_k < C^*_i$ *or there exists a* j *where* $r_j < r_q$ *and* $C^*_j > C^*_q\}$

e. $\Delta > \hat{S}^*_i(r_k) - r_k$, *when* $S^*_k < C^*_i$

f. $\Delta \geq \max_{q\in N}\{R_i(r_q) - p_q \mid S^*_i < r_q < C^*_i \leq S^*_q\}$, *when* $S^*_k < C^*_i$.

Proof. The specifications of strict or weak inequalities in parts b, c, and f are due to the tie breaking rule that we choose.

($\Leftarrow$). For each part, we describe why π^* is no longer optimal when the stated condition is violated.

a. A piece of job k is processed before S^*_i.

b. Job k preempts job j, where we allow $j = i$.

c. Some job u, where $S^*_q < S^*_u < C^*_q$, no longer preempts job q. However, if there exists some $j \in \mathcal{E}$ such that $C^*_j \leq S^*_q$ and $S^*_j - r_j < R_q(r_u) - p_u$, then job j blocks job q from moving earlier enough to change π^*.

d. Some job $q \in \mathcal{E}$ starts earlier in σ' by an amount that is less than Δ. This creates a gap of size $r_q - S^*_q - \Delta > 0$ before q in the schedule. Some job j such that $R_j(r_q) > 0$ can be inserted in this gap. Note that job q does not preempt any jobs in σ^*, because $q \in \mathcal{E} \cup \{e\}$. Thus, job j is not processed immediately before job q in σ^*. Hence, π^* changes. If $C^*_i > S^*_k$, then job i can be processed in the gap before job q.

e. The piece of i that starts at $\hat{S}^*_i(r_k)$ is replaced by job k.

f. Since the remaining processing time of job i at S^*_k is increased, i is no longer selected over job q which is released after S^*_i and which starts after C^*_i.

($\Rightarrow$). We establish that if the conditions in parts a – f are satisfied, then π^* is optimal for the modified problem. The jobs are partitioned into three subsets: those that start before S^*_i, job i, and those that start after S^*_i. Note that the sequence after job e does not change, because e starts at the same time in σ^* and σ'. Also from Lemma 4, the jobs processed before r'_k do not change when $S^*_k < C^*_i$.

For the jobs that start before S^*_i, there are three cases. First, if $S^*_k \geq C^*_i$, then the condition in part a prevents k from starting during idle time before S^*_i. Second, if $S^*_k \geq C^*_i$, then the condition in part b prevents job k from preempting i or a job that is processed before i. Finally, if $S_k^* < C_i^*$, then the condition in part e prevents k from delaying the piece of i that starts at $\hat{S}_i^*$.

For job i, there are three cases. First, if $S_k^* \geq C_i^*$, then the condition in part b prevents k from delaying the processing of i. Alternatively, if $S_k^* < C_i^*$, then k can start earlier. The condition in part e prevents k from delaying the last piece of i before S_k^*. The last possibility is that some job q displaces i because the remaining processing time of i has increased. The condition in part f prevents this situation.

For the jobs that start after S_i^*, there are two cases. First, the condition in part c ensures that a job u which preempts a job $q \in \mathcal{E}$ in σ^* continues to do so. Second, some job $q \in \mathcal{E} \cup \{e\} \setminus \{k\}$ may not start early enough to prevent idle time in the schedule. The condition in part d prevents any job from being processed during this idle time period. □

Lemma 6 *For some $k \in N$, let $r'_k = r_k + \Delta$ be a parameter change for $1|r_j, pmtn|\sum C_j$. Suppose $\Delta > 0$ and suppose i is the last job that is processed before r_k in σ^*. Then, π^* is an optimal sequence for the modified problem if and only if*

a. $\Delta \leq \max\Big\{S_k^*, \min_{j \in N}\{r_j \mid R_j(S_k^*) > 0\}\Big\} - r_k$, *when* $S_k^* \geq C_i^*$

b. $\Delta \leq S_k^* - r_k + \min_{q \in \mathcal{L}, u \in N}\Big\{I(r_k, r_q) + \max\{r_u - C_q^*, 0\} + p_u - R_q(r_u) \mid C_q^* \leq S_u^*,\ p_u < p_q\Big\}$

c. $\Delta < S_k^* - r_k + \min_{q \in \mathcal{L}}\{I(r_k, r_q) + \alpha_q(r_q)\}$

d. $\Delta < \alpha_i(C_k^*)$, *when* $S_k^* < C_i^*$

e. $\Delta < \min_{q \in \mathcal{L}}\{R_i(r_q) - p_q \mid S_k^* \leq S_q^* < C_i^*\}$, *when* $S_k^* < C_i^*$.

Proof. Similar to that of Lemma 5. □

We provide sufficient conditions for π^* to remain optimal after a single release date change. That is, we describe conditions under which the SRPT rule selects the same job sequence for the original data and the modified data. In part i of Theorem 6, job k does not preempt job i in σ^*. In part ii, job k preempts job i at r_k in σ^*. From the SRPT rule, job k does not preempt job i at any time other than r_k. Therefore, these two cases are exhaustive.

Theorem 6 *For some $k \in N$, let $r'_k = r_k + \Delta$ be a parameter change for $1|r_j, pmtn|\sum C_j$. Suppose that i is the last job that is processed before r_k in σ^*. Then, π^* is optimal for the modified problem if and only if one of the following conditions holds for an arbitrarily small $\epsilon > 0$.*

i. $S_k^* \geq C_i^*$ *and*

$\max\Big\{\min\{t < S_i^* \mid I(t, S_i^*) = 0\} - r_k, \quad \max\{t \mid R_j(t) \geq p_k \textit{ for some } j \textit{ where } C_j^* \leq S_k^*\} - r_k,$
$\max_{q\in\mathcal{E}} \max_{u\in N}\{p_u - R_q(r_u) \,|\, S_q^* < S_u^* < C_q^*, \textit{ no } j \in \mathcal{E} \textit{ exists such that } C_j^* \leq S_q^* \textit{ and } S_j^* - r_j <$
$R_q(r_u) - p_u\} + \epsilon, \max_{q\in\mathcal{E}\cup\{e\}\setminus\{k\}}\{r_q - S_q^* \mid \textit{there exists a } j \textit{ where } r_j < r_q \textit{ and } C_j^* > C_q^*\}\Big\}$
$\leq \Delta \leq \min\Big\{\max\{S_k^*, \min_{j\in N}\{r_j \mid R_j(S_k^*) > 0\}\} - r_k,$
$S_k^* - r_k + \min_{q\in\mathcal{L}, u\in N}\{I(r_k, r_q) + \max\{r_u - C_q^*, 0\} + p_u - R_q(r_u) \mid C_q^* \leq S_u^*, \; p_u < p_q\},$
$S_k^* - r_k + \min_{q\in\mathcal{L}}\{I(r_k, r_q) + \alpha_q(r_q)\} - \epsilon\Big\}.$

ii. $S_k^* < C_i^*$ *and*

$\max\Big\{\max_{q\in\mathcal{E}} \max_{u\in N}\{p_u - R_q(r_u) \,|\, S_q^* < S_u^* < C_q^*, \textit{ no } j \in \mathcal{E} \textit{ exists such that } C_j^* \leq S_q^* \textit{ and } S_j^* -$
$r_j < R_q(r_u) - p_u\} + \epsilon, \quad \max_{q\in\mathcal{E}\cup\{e\}\setminus\{k\}}\{r_q - S_q^*\}, \quad \hat{S}_i^*(r_k) - r_k + \epsilon,$
$\max_{q\in N}\{R_i(r_q) - p_q \mid S_i^* < r_q < C_i^* \leq S_q^*\}\Big\} \leq \Delta \leq$
$\min\Big\{S_k^* - r_k + \min_{q\in\mathcal{L}, u\in N}\{I(r_k, r_q) + \max\{r_u - C_q^*, 0\} + p_u - R_q(r_u) \mid C_q^* \leq S_u^*, \; p_u < p_q\},$
$S_k^* - r_k + \min_{q\in\mathcal{L}}\{I(r_k, r_q) + \alpha_q(r_q)\} - \epsilon, \quad \alpha_i(C_k^*) - \epsilon, \min_{q\in\mathcal{L}}\{R_i(r_q) - p_q \,|\, S_k^* \leq S_q^* < C_i^*\} - \epsilon\Big\}.$

Proof. The results follow from Lemmas 5 and 6. □

We now present a result that addresses issue I3 for Q3. This result illustrates the usefulness of finding bounds on parameter changes that maintain the job sequence. A substantial parameter change may be needed to alter the optimal sequence, while only a small parameter change may alter the optimal schedule. As a consequence, the ranges in the conditions of Theorem 6 are wider than similar ranges for the optimal schedule.

Theorem 7 *If either of the two conditions in the statement of Theorem 6 is satisfied, then a new optimal schedule can be found in* $O(n)$ *time.*

Proof. The optimality of the new schedule follows from Lemma 3 and Theorem 6. Given σ^*, the conditions in Theorem 6 can be verified in $O(n)$ time. □

Observe that the SRPT algorithm of Miller and Schrage (1966) requires $O(n \log n)$ time to reconstruct an optimal schedule after the change in data. Thus, the use of the optimal job sequence permits a reduction in the time required to find a new optimal schedule.

3.3 Linear Programming Algorithms

Consider problems that are solved efficiently by some variant of linear programming. In general, the dual solution provides sensitivity analysis information. However, the linear programming formulation may obscure the answer to some sensitivity analysis questions. For instance, consider $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$. This problem can be efficiently formulated and solved as an assignment problem. The formulation uses decision variables x_{kt}, where $x_{kt} = 1$ if job k is processed at time t, and $x_{kt} = 0$ otherwise. The cost of completing job k at time t is $v_{kt} = w_k t$. If we change the weight of job k, then all of the objective function coefficients $v_{kr_k+1}, \ldots, v_{k\bar{d}_k}$ change. This is an example of issue I2 where simultaneous changes occur due to the modelling process. Since these changes have a specific structure, we can improve on standard linear programming sensitivity analysis for this problem.

First, we address issue I2 by answering Q4.3 for simultaneous changes to several weights. We begin with two definitions (see Ahuja *et al.*, 1993). Given a set of flows in a graph, the associated *residual network* is an identical graph, but with reduced arc capacities that represent the amount of additional flow that can be sent along each arc. The *node potentials* are the dual linear programming variables associated with the flow balance equations.

An efficient method for solving the assignment problem is the Cost Scaling Algorithm of Gabow and Tarjan (1989). Given integer data, this algorithm runs in $O(n^{2.5}\log(nc))$ time, where c is the maximum cost. For our problem, $c = \max_{1\le j\le n}\{w_j \bar{d}_j\}$. We also assume integer data.

Theorem 8 *Let $w'_k = w_k + \Delta_k$, where Δ_k is integral, for $k = 1, \ldots, n$, be parameter changes for $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$. Then, σ' can be found in $O(n^{2.5}\log(n \max_{1\le k\le n}\{|\Delta_k|\bar{d}_k\}))$ time.*

Proof. Consider the residual network associated with an optimal solution for the original data set. Let λ_{ik} denote the reduced cost of arc (i, k) for the optimal node potential vector. Because the solution is optimal, $\lambda_{ik} \ge 0$ for every arc (i, k) in the residual network. The parameter changes imply that the absolute value of each cost in the assignment formulation changes by no more than $\max_{1\le k\le n}\{|\Delta_k|\bar{d}_k\}$. Thus, after the parameter changes, $\lambda_{ik} \ge -\max_{1\le k\le n}\{|\Delta_k|\bar{d}_k\}$ for all arcs (i, k) in the residual network. Therefore, we can apply additional iterations of the cost scaling algorithm, which finds a new optimal schedule in the stated time. □

If the original optimal solution is not used, then resolving the scheduling problem by the Cost Scaling Algorithm requires $O(n^{2.5}\log(n\max_{1\le k\le n}\{(w_k+|\Delta_k|)\bar{d}_k\}))$ time to answer Q4.3. This time requirement may be substantially more than that in Theorem 8. Therefore, Theorem 8 is an example of issue I2.

We now consider a single parameter change of the form $w'_k = w_k + \Delta$, for some $k \in N$. This change increases the assignment costs from $v_{kt} = w_k t$ to $v'_{kt} = (w_k+\Delta)t$, for $t = r_k+1,\ldots,\bar{d}_k$. We formulate this problem as a matching problem on a bipartite graph. There are several closely related sensitivity analysis studies of the matching problem on a general graph. Weber (1981) and Derigs (1985) consider sensitivity analysis following the deletion of, or a weight change to, a single arc. A related problem is considered by Ball and Taverna (1985). Motivated by issue I2, we propose the following algorithm to answer Q4.3.

Algorithm WCCHG

0. Let σ^* be an optimal solution for $1|r_j,\bar{d}_j,p_j=1|\sum w_jC_j$ formulated as an assignment problem. Solution σ^* assigns processing time C^*_j to job j, for $j=1,\ldots,n$. Let k denote the job that has a modified weight.
1. Change the assignment costs from v_{kt} to $v'_{kt} = (w_k+\Delta)t$, for $t=r_k+1,\ldots,\bar{d}_k$. Let $\hat{\sigma}$ be σ^* with job k and time C^*_k removed.
2. Starting with solution $\hat{\sigma}$, reoptimize the solution, where job k and time C^*_k are included, by using one augmenting path iteration of the Hungarian Algorithm (Kuhn, 1955).

Theorem 9 *For some $k\in N$, let $w'_k = w_k+\Delta$ be a parameter change for $1|r_j,\bar{d}_j,p_j=1|\sum w_jC_j$. Then, Algorithm WCCHG finds a new optimal solution in $O(n^2)$ time.*

Proof. If $\hat{\sigma}$ is not optimal for the assignment subproblem defined by jobs $\{1,\ldots,n\}\setminus\{k\}$ and time periods $\cup_{1\le j\ne k\le n}\{r_j+1,\ldots,\bar{d}_j\}\setminus\{C^*_k\}$, then σ^* is not optimal for the original problem. In Step 2, we find an augmenting path of solution $\hat{\sigma}$ with minimal cost. The optimality of the solution follows from Papadimitriou and Steiglitz (1982).

Step 1 requires $O(n)$ time. Since job k is processed in one of the time periods $\{r_k+1,\ldots,\min\{r_k+n,\bar{d}_k\}\}$, Step 2 requires $O(n^2)$ time (Papadimitriou and Steiglitz, 1982). Thus, the overall time complexity of the sensitivity analysis in WCCHG is $O(n^2)$. $\square$

Answering Q4.3 with an efficient strongly polynomial time assignment algorithm (Ahuja *et al.*, 1993) requires $O(n^3)$ time. Thus, Theorem 9 is an example of issue I2. Using the current optimal solution and Steps 1 and 2 of WCCHG, Q3 can be answered in less time than it takes to find the optimal schedule. Note that Karp and Orlin (1981) describe parametric shortest path algorithms for changes in cost that are similar to those described in Step 1 of WCCHG.

Corollary 1 *For some $k \in N$, let $r'_k = r_k + \Delta$ or $\bar{d}'_k = \bar{d}_k + \Delta$ be a parameter change for $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$. Then, a procedure similar to Algorithm WCCHG finds a new optimal solution in $O(n^2)$ time.*

We now consider a single parameter change of the form $w'_k = w_k + \Delta$ for some $k \in N$. This change may alter several of the $v_{kt} = w_k t$ costs (issue I2). Since $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$ can be formulated as a linear program, standard linear programming sensitivity analysis results are first considered. We provide conditions that are sufficient to maintain optimality.

Let S denote the set of times at which processing occurs in any feasible schedule that does not have inserted idle time, where $|S| = n$. When there are n jobs and n times, a linear programming assignment problem has $2n$ constraints. However, the constraint matrix does not have full row rank. A standard formulation eliminates one row, which makes the remaining rows linearly independent (for example, see Gal, 1979). As a result, there are $n-1$ basic variables with a value of zero. We construct an equivalent formulation in which all of the basic variables corresponding to jobs $1, \ldots, n$ have a value of 1. We introduce a dummy job 0 and a dummy time -1. The new problem has $(n+1)^2$ variables and $2n+2$ constraints. The cost of scheduling job 0 is 0 at time -1 and is large for all of the other n times. Thus, only job 0 is scheduled at time -1. We augment S to include time -1.

Delete the row $\sum_{t \in S} x_{0t} = 1$ to make the remaining rows linearly independent. Let A be the resulting matrix. Arrange the assignment constraints such that they appear in the order

$$\begin{aligned} \sum_{j=0}^{n} a_{jt} x_{jt} &= 1, \quad t \in S \\ \sum_{t \in S} a_{kt} x_{kt} &= 1, \quad k = 1, \ldots, n. \end{aligned}$$

Let A_{jt} denote the column of the coefficient matrix corresponding to variable x_{jt} for $j \in \{0, \dots, n\}$ and $t \in S$. Each A_{jt} has at most two ones and the remaining entries are zero. Let x^* denote an optimal solution to $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$. Consider the basis B constructed from the columns A_{0t}, $t \in S$, and A_{kt} where $x^*_{kC^*_k} = 1$ and $k = 1, \dots, n$. Observe that this basis is optimal. The first $n+1$ columns of B form an identity matrix above a matrix containing all zero entries. Each of the last n columns has exactly two entries of 1, one along the diagonal and one corresponding to the row associated with the time at which the job is completed. Let $(B^{-1}A_{kt})_k$ be the component of $B^{-1}A_{kt}$ corresponding to job k. Let $\lambda_{kt} \geq 0$ denote the reduced cost associated with nonbasic variable x^*_{kt}. By definition, this is the maximum decrease in the cost of the associated variable such that it remains nonbasic, assuming that no other parameter changes occur.

Suppose that we use either the Hungarian Algorithm or the Cost Scaling Algorithm to solve $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$. Both algorithms provide node potentials α_k and β_t for $t = r_k + 1, \dots, \bar{d}_k$, at optimality. Then, we can compute $\lambda_{kt} = v_{kt} - \alpha_k - \beta_t$ for a nonbasic variable in constant time for each $k \in N$ and $t \in S$. The next result addresses issue I2 by answering Q4.1.

Theorem 10 *Consider the optimal solution and the reduced costs λ_{kt} for $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$ formulated as an assignment problem. For some $k \in N$, let $w'_k = w_k + \Delta$ be a parameter change for $1|r_j, \bar{d}_j, p_j = 1|\sum w_j C_j$. Schedule σ^* remains optimal if*

$$\underline{\Delta}_k = \max_{t=C^*_k+1,\dots,\bar{d}_k} \Big\{\lambda_{kt}/(C^*_k - t)\Big\} \leq \Delta \leq \overline{\Delta}_k = \min_{t=r_k+1,\dots,C^*_k-1} \Big\{\lambda_{kt}/(C^*_k - t)\Big\}.$$

Moreover, the values $\underline{\Delta}_k$ and $\overline{\Delta}_k$, for $k = 1, \dots, n$, can be found in $O(n^2)$ time.

Proof. We first consider $\overline{\Delta}_k$. From the discussion preceding this theorem, $x^*_{kt} = 0$ for $t \in \{r_k + 1, \dots, C^*_k - 1, C^*_k + 1, \dots, \bar{d}_k\}$. Also, since $v_{kt} = w_k t$, when w_k increases, scheduling job k at time t becomes more costly relative to other jobs. Thus, if σ^* is no longer optimal because of an increase in w_k, then x_{kt} becomes basic for some $t \in \{r_k + 1, \dots, C^*_k - 1\}$. We only need to find the value of Δ for which $\lambda_{kt} - (-\Delta t) - \Delta C^*_k (B^{-1}A_{kt})_k = 0$ (see, for example, Gal, 1979, or Hillier and Lieberman, 1995). By definition, when $\Delta \leq \overline{\Delta}_k$, the optimal solution does not change. Hence, a sufficient condition for σ^* to remain optimal is

$$\Delta \leq \overline{\Delta}_k = \min_{t=r_k+1,\dots,C^*_k-1} \Big\{\lambda_{kt}/[C^*_k (B^{-1}A_{kt})_k - t]\Big\}. \tag{2}$$

For a given k, consider the calculation of $\overline{\Delta}_k$ in (2). The matrix B^{-1} is the matrix B where each nonzero off diagonal entry in B is replaced by -1. Also, each column of A_{kt} and each row of B^{-1} has at most two nonzero entries. The first nonzero entry of A_{kt} is associated with a coefficient in the equation $\sum_{j=0}^{n} a_{0t}x_{0t} = 1$ and the second nonzero entry is associated with a coefficient in the equation $\sum_{t\in S} a_{kt}x_{kt} = 1$. Since all entries below the main diagonal of B^{-1} are 0, it follows that $(B^{-1}A_{kt})_k = 1$. Substituting into (2) establishes the value of $\overline{\Delta}_k$.

Because there are $O(n)$ times t to consider, the computational work required to compute $\overline{\Delta}_k$ is $O(n)$. Thus, the total computational work requires $O(n^2)$ time because there are n jobs.

A similar analysis establishes the bound for $\underline{\Delta}_k$. □

Since all parameter changes are in the same direction, the sensitivity range presented in Theorem 10 is wider than that given by the 100% rule (Bradley *et al.*, 1977) and by linear programming analysis for a single cost change.

4 Planning for Sensitivity Analysis

In many situations, sensitivity analysis is considered only after an optimal solution is found. However, the optimization approach taken and the computational effort required may change considerably if sensitivity analysis is anticipated (issue I8). Further, relevant sensitivity analysis questions and the appropriate procedure may depend on the parameters being modified. In this section, first we discuss situations where we should spend more time during the solution procedure to speed up subsequent sensitivity analysis. Then, the selection is considered of a different and more complicated solution procedure to facilitate subsequent sensitivity analysis. Next, we examine an example where the particular optimization procedure depends on the specific type of sensitivity analysis to be performed. Finally, we study particular solutions that are robust to certain types of sensitivity analysis.

First, as an example of issue I8, we discuss a situation where optimization is performed under less time pressure, or with less cost, than sensitivity analysis. Consider a production shop where the schedule for the next day can be run overnight. Time constraints during the night are less severe than those arising during the following day's production process. On the following day, we may need a quick evaluation of the effect of parameter changes on the optimal cost or optimal schedule. Consequently, we may be willing to use a less efficient algorithm overnight, if it allows us to answer sensitivity questions more quickly during the following day.

Theorem 11 *For some $k \in N$, let $p'_k = p_k + \Delta_1$ and $w'_k = w_k + \Delta_2$ be parameter changes for $1||\sum w_j C_j$. If π^* is no longer optimal, then π' and z'' can be found in $O(\log n)$ time.*

Proof. Reindex the jobs such that $w_1/p_1 \geq \cdots \geq w_n/p_n$. If π^* is no longer optimal, then from the WSPT rule, either $w'_k/p'_k > w_{k-1}/p_{k-1}$ or $w'_k/p'_k < w_{k+1}/p_{k+1}$. Assume that $w'_k/p'_k > w_k/p_k$. The other case is similar. Let $B_1 = \{j \mid w_j/p_j > w'_k/p'_k\}$, $B_2 = \{j \mid w'_k/p'_k \geq w_j/p_j \geq w_k/p_k\}$ and $B_3 = \{j \mid w_k/p_k > w_j/p_j\}$. Using binary search over the integers in the interval $[0, n - |B_3|]$, we can find $|B_1|$ in $O(\log n)$ time. Placing job k in position $|B_1| + 1$ in the sequence, and moving job j one position later for all $j \in B_2$ creates a new optimal sequence (see Theorem 5). To represent this new sequence in constant time, we store σ^* in a doubly linked list and adjust only the links for job k and jobs adjacent to it. The resulting change in cost is

$$p_k \sum_{j \in B_2} w_j + (p'_k - p_k) \sum_{j \in B_3} w_j + w'_k(\sum_{j \in B_1} p_j + p'_k) - w_k(\sum_{j \in B_1} p_j + \sum_{j \in B_2} p_j + p_k).$$

The change in cost can be computed in constant time if $\sum_{j=i}^{n} w_j$ and $\sum_{j=1}^{i} p_j$ for $i = 1, \ldots, n$, are precomputed during optimization. □

To reduce the computational work of sensitivity analysis, the key idea is to precompute the partial sums $\sum_{j=i}^{n} w_j$ and $\sum_{j=1}^{i} p_j$ for $i = 1, \ldots, n$, during optimization. Observe that the time complexity of the optimization procedure remains unchanged, since $\sum_{j=i}^{n} w_j$ and $\sum_{j=1}^{i} p_j$ for $i = 1, \ldots, n$ can be computed in $O(n)$ time. However, the complexity of finding a new optimal sequence and its cost, to answer Q4.2 and Q4.3, is reduced from $O(n)$ to $O(\log n)$ time.

Again motivated by issue I8, the question of when it may be worthwhile to use a different optimal procedure is examined for $1||\sum U_j$. The conventional way to solve this problem is by using the algorithm of Moore (1968). Because the jobs must be sorted in EDD order, the time complexity of this algorithm is $O(n \log n)$. To answer Q1, we propose a dynamic programming optimization procedure that requires $O(n^2)$ time. At the completion of optimization, we have the information necessary to perform sensitivity analysis.

Since the on-time jobs for $1||\sum U_j$ can be sequenced in EDD order in an optimal schedule, we reindex the jobs such that $d_1 \leq \cdots \leq d_n$. Consider a network, G, consisting of nodes defined by the pair (i,j), where exactly j jobs from the set $\{1,\dots,i\}$ are on time. In G, the only arcs from node $(i-1,j-1)$ are those to nodes $(i,j-1)$ and (i,j), with distances 0 and p_i, respectively. The arc from $(i-1,j-1)$ to $(i,j-1)$ represents a decision that job i is late, and the arc from $(i-1,j-1)$ to (i,j) represents a decision that job i is on time. The shortest path from $(0,0)$ to (i,j) is the minimum makespan required to schedule exactly j of jobs $1,\dots,i$ on time.

The following forward dynamic program computes this makespan. Let

$$\begin{aligned} f_i(j) &= \text{minimum makespan required for } j \text{ jobs from the set } \{1,\dots,i\} \text{ to be on time} \\ &= \begin{cases} \min\{f_{i-1}(j),\ f_{i-1}(j-1)+p_i\}, & \text{if } \min\{f_{i-1}(j),\ f_{i-1}(j-1)+p_i\} \le d_i \\ \infty, & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

The boundary conditions are, for $i=0,\dots,n$ and $j \le i$,

$$f_i(j) = \begin{cases} 0, & \text{if } i=0 \text{ and } j=0 \\ \infty, & \text{otherwise.} \end{cases}$$

The optimal value of the recursion is $f^* = \max\{j \mid f_n(j) < \infty\}$, where $z^* = n - f^*$.

Next, we solve a backward recursion. The value function is

$$\begin{aligned} g_i(j) &= \text{latest time at which } j \text{ jobs from } \{1,\dots,i\} \text{ can complete on time,} \\ &\quad\ \text{such that at least } f^* \text{ jobs from } \{1,\dots,n\} \text{ can be on time} \\ &= \min\Big\{d_i, \max\{g_{i+1}(j), g_{i+1}(j+1)-p_{i+1}\}\Big\}. \end{aligned} \tag{4}$$

The boundary conditions are, for $i=0,\dots,n$ and $j \le i$,

$$g_i(j) = \begin{cases} d_i, & \text{if } i=n \text{ and } j=f^* \\ -\infty, & \text{otherwise.} \end{cases}$$

We now construct an algorithm to answer Q1 for a parameter change of the form $p'_k = p_k + \Delta$.

Algorithm DPSLACK

```
0. Reindex the jobs such that $d_1 \le \cdots \le d_n$.
1. For $j \le i$, solve (3) for $i = 1,\dots,n$, and (4) for $i = n-1,\dots,0$.
2. For $i = 1,\dots,n$,
      If $\max_{j=0,\dots,i-1}\{g_i(j) - f_{i-1}(j)\} \ge 0$, then set $\overline{\Delta}_i = \infty$.
      Otherwise, set $\overline{\Delta}_i = \max_{j=0,\dots,i-1}\{g_i(j+1) - f_{i-1}(j) - p_i\}$.
3. For $i = 1,\dots,n$, set $\underline{\Delta}_i = \max\Big\{0, \max_{j=0,\dots,i-1}\{g_i(j) - f_{i-1}(j)\}\Big\} - p_i$.
4. Output $\overline{\Delta} = (\overline{\Delta}_1,\dots,\overline{\Delta}_n)$ and $\underline{\Delta} = (\underline{\Delta}_1,\dots,\underline{\Delta}_n)$.
```

Theorem 12 *For some $k \in N$, let $p'_k = p_k + \Delta$ be a parameter change for $1||\sum U_j$. Then, Algorithm DPSLACK finds $\overline{\Delta}$ and $\underline{\Delta}$ for z^*.*

Proof. We first consider $\overline{\Delta}$. The computation of $\max_{j=0,\ldots,i-1}\{g_i(j)-f_{i-1}(j)\}$ in Step 2 determines whether there is a feasible path from node $(0,0)$ to node (n,f^*) for which job k is not scheduled on time. That is, does there exist a j such that the arc from node $(k-1,j)$ to node (k,j) is on the feasible path? If so, then an increase of any size in p_k does not change the value of the optimal schedule since job k is not scheduled. Thus, $\overline{\Delta}_k = \infty$. Otherwise, every optimal schedule for the original data set schedules job k on time. In this case, the second computation in Step 2 calculates the maximum increase in p_k such that a feasible path still exists, i.e. such that the optimal value remains unchanged. This maximum increase is $\overline{\Delta}_k$. Note that $g_i(j+1) - f_{i-1}(j) \geq p_i$ for any arc from node $(i-1,j)$ to node $(i,j+1)$ on a feasible path from node $(0,0)$ to node (n,f^*).

Next, we consider $\underline{\Delta}$. Observe that, given any feasible schedule with f^*+1 on time jobs where p_k has been reduced by $|\underline{\Delta}|$, removing job k creates a feasible schedule with f^* on-time jobs. Hence, we find a schedule with f^* jobs on time and job k late, where $g_k(j) - f_{k-1}(j)$ is as large as possible. This provides the minimum time to process job k. Note that $g_i(j) - f_{i-1}(j) < p_i$ for any arc from node $(i-1,j)$ to node (i,j) on a feasible path from node $(0,0)$ to node (n,f^*); otherwise, we can process job i on time. This creates a feasible solution with f^*+1 on time jobs. When $g_k(j) - f_{k-1}(j) \leq 0$ for all j, every feasible schedule with f^* on-time jobs requires job k to be on time. Thus, there is no feasible schedule with f^*+1 on-time jobs where $p'_k > 0$. □

Note that the computations in Steps 2 and 3 are similar to the computations of the slack on an arc in the Critical Path Method for project management.

The use of DPSLACK instead of Moore's algorithm arises from the need to answer Q1. It requires $O(n^2)$ time to compute (3) and (4) in Step 1. An alternative procedure using Moore's algorithm proceeds as follows. For each of the $O(n)$ jobs, a binary search procedure finds the smallest increase and the smallest decrease in the processing time that change z^*. At each value in the binary search procedure, Moore's algorithm requires $O(n \log n)$ time.

The last example of issue I8 relates to a situation where the choice of optimization approach is motivated by sensitivity analysis requirements. Consider $1||\sum w_j U_j$, which is binary NP-hard (Karp, 1972), but solvable in pseudopolynomial time by dynamic programming (Lawler and Moore, 1969). Two alternative dynamic programming algorithms are available. The first algorithm uses a state space defined by (job, schedule length) pairs, arc costs defined by job weights, and a value function that minimizes the total weight of late jobs. This algorithm requires $O(n \sum p_j)$ time. The second algorithm uses a state space defined by (job, total weight of late jobs) pairs, arc costs defined by job processing times, and a value function that minimizes the schedule length. This algorithm requires $O(n \sum w_j)$ time. If $\sum p_j$ and $\sum w_j$ are of approximately the same order of magnitude, then the running times of the two algorithms are similar. However, anticipating the need for a specific type of sensitivity analysis might justify the choice of one dynamic program over another. For example, using the first algorithm for optimization, an application of DPSLACK answers Q1 for a change in a weight, but not for a change in a processing time. This is because the arc costs are weights. A similar procedure that uses the second algorithm for optimization answers Q1 for a change in a processing time, but not for a change in a weight. This is because the arc costs are processing times. Therefore, the type of sensitivity analysis required might influence the type of optimization procedure used.

We now consider the issue of selecting among optimal schedules based on their robustness (issue I9). This is important in planning for a parameter change. There are a variety of scheduling problems where multiple optimal schedules occur frequently. The choice of an optimal schedule may greatly affect the robustness of sensitivity analysis results. An example is $P|pmtn|C_{\max}$. An optimal procedure schedules jobs in arbitrary index order on the lowest index machine with spare capacity (McNaughton, 1959). Since each ordering of the indices provides a different optimal schedule, there are many optimal schedules. We choose among the optimal schedules by finding one that is most robust with respect to π^*, i.e. provides the widest range for Q1.

For all $k \in N$, the change in processing time, Δ_k, satisfies $-p_k \leq \Delta_k < \infty$. For a specified k, the following procedure finds an optimal schedule for $P|pmtn|C_{\max}$, where the job sequence remains optimal for any $\Delta_k \in [-p_k, \infty)$. Thus, the maximum optimality range is achieved.

Algorithm WIDE

```
0. Input $p_1, \ldots, p_n$, job $k$, and the number of machines $m$.
1. Find $l = \text{argmax}\{p_j \mid j \in N \setminus \{k\}\}$ and $C_{\max} = \max\{\sum_{j=1}^n p_j/m, p_l\}$.
   Divide job $k$ into $m$ pieces, each with processing time $p_k/m$. Start processing the $i$th piece
```

at time $s_i = (i-1)(C_{\max} - p_k/m)/(m-1)$ on machine M_i, for $i = 1, \ldots, m$.

2. Let $R = \{j \in N \setminus \{k\} \mid p_j > C_{\max} - p_k/m\}$.

   If $C_{\max} = \sum_{j=1}^{n} p_j/m$ or $C_{\max} = p_k$, then go to Step 4.

3. Process no job during the interval $\Big[s_i + p_k/m, s_i + p_k/m + \min\{(C_{\max} - p_k)/(m-1), C_{\max} - (\sum_{j=1}^{n} p_j - p_l)/(m-1)\}\Big]$ on machine M_i, for $i = 1, \ldots, m-1$.

4. Schedule the jobs in R. Then, schedule the remaining jobs. Start processing each job in index order on the available machine with the lowest index as early as possible (as in McNaughton's Algorithm). If necessary, complete processing on the machine with the next higher index. Do not process a job when it is being processed on the previous machine.

While π^* remains unchanged, the following procedure adjusts σ^* after p_k changes by Δ_k.

Procedure PKADJ

1. If $\Delta_k < 0$, then reduce the processing time of each piece of job k by Δ_k/m. Also, decrease the start times of all jobs processed on the same machine after a given piece of k by $\min\{\Delta_k/m, t\}$, where a decrease of more than t results in simultaneous processing of a job on two machines.

2. If $\Delta_k > 0$ and $C_{\max} = \sum_{j=1}^{n} p_j/m$, then increase by an equal amount the processing times of all pieces of job k on all machines until a total of $\min\{\Delta_k, m(C_{\max} - p_k)/(m-1)\}$ is added. Also, increase by an equivalent amount the start times of all the jobs that are processed on the same machine after a given piece of k. Add any remaining processing time to the piece of job k on machine M_m.

3. If $\Delta_k > 0$ and $C_{\max} > \sum_{j=1}^{n} p_j/m$, then increase by equal amounts the processing times of all the pieces of job k on machines $M_1, \ldots, M_{m-1}$. Continue until $v = \min\{\Delta_k, C_{\max} - p_k, (m-1)C_{\max} - \sum_{j=1}^{n} p_j + p_l\}$ total processing time is added.

   a. When $v = C_{\max} - p_k$, add any remaining processing time to the piece of job k on M_m.

   b. When $v = (m-1)C_{\max} - \sum_{j=1}^{n} p_j + p_l$, increase the processing time of each piece of job k by the same amount on all machines until a further total processing time of $\min\{\Delta_k - v, m(C_{\max} - p_k - v)/(m-1)\}$ is added. Also, increase by an equivalent amount the start times of all jobs that are processed on the same machine after a given piece of k. Add any remaining processing time to the piece of job k on M_m.

In Step 2 of PKADJ, when the amount of processing added to job k is $m(C_{\max}-p_k)/(m-1)\}$, then $p'_k = C_{\max}$. Any additional processing of a piece of k on one of the machines $M_1, \dots, M_{m-1}$ results in simultaneous processing of k on two machines. Consequently, the remaining processing time is added to the piece of job k on machine M_m. While the makespan increases, the sequence remains optimal because $p'_k = C_{\max}$.

In Step 3 of PKADJ, when $\Delta_k > 0$ and $C_{\max} > \sum_{j=1}^{n} p_j/m$, the first stage inserts additional processing time into the idle time intervals created in Step 3 of WIDE on machines $M_1, \dots, M_{m-1}$. If $v = C_{\max} - p_k$, then the total processing time of jobs $N \setminus \{l\}$ is small enough that p'_k becomes equal to $C_{\max}$ at least as soon as any of the first $m-1$ machines processes jobs for $C_{\max}$ units of time. Any remaining processing time of $\Delta_k - (C_{\max} - p_k)$ is added to the piece of k on machine M_m. If $v = (m-1)C_{\max} - \sum_{j=1}^{n} p_j + p_l$, then idle time of size $C_{\max} - (\sum_{j=1}^{n} p_j - p_l)/(m-1)$ is inserted on each of the first $m-1$ machines in Step 3 of WIDE. After the first part of Step 3 of PKADJ, all this idle time is filled by processing pieces of k. Also, all machines complete at $C_{\max}$. Then, an equal amount of processing time is added to the piece of k on each of the m machines until the total processing time of k on the first $m-1$ machines is $C_{\max} - p_k/m$. Since the piece of k that is processed on machine M_m starts processing at time $C_{\max} - p_k/m$, job k is processed throughout the entire processing time interval. Any remaining processing time of $\Delta_k - m(C_{\max} - p_k - v)/(m-1)\}$ is added to the piece of job k on machine M_m.

Theorem 13 *For some $k \in N$, let $p'_k = p_k + \Delta_k$ be a parameter change for $P|pmtn|C_{\max}$. Then, in $O(m+n)$ time, Algorithm WIDE finds an optimal schedule for which π^* remains optimal for $\Delta_k \in [-p_k, \infty)$.*

Proof. WIDE assigns each job $j \in N \setminus \{k\}$ to at most two machines, where those machines have consecutive indices.

We first consider the case when $\Delta_k < 0$. If we reduce each piece of job k in PKADJ, then the job pieces that are processed after a piece of k on any given machine all start earlier by the same amount. Also, on any given machine, the job pieces that are processed before a piece of k do not change processing times. Suppose job j is processed on machines M_i and M_{i+1}. Since the jobs in R are assigned first in WIDE, no processing of job j occurs on M_{i+1} after the completion of the piece of k on M_i. Further, from the definition of R and Step 4 of WIDE, each job $j \in R$ starts on M_i no later than the start time of the piece of k on M_{i+1}. Hence, the time between when j finishes processing on M_{i+1} and a piece of j starts after the piece of k on M_i is at least $C_{\max} - p_j$. On each machine, the completion time becomes smaller by an equal amount until either $p'_k = 0$ or the makespan is equal to the processing time of some job. The scheduling procedure in Step 1 of PKADJ ensures that no pair of jobs is processed simultaneously. Consequently, the sequence remains feasible. Because either each machine has the same amount of processing or the makespan is equal to the processing time of some job, the sequence remains optimal.

For the case when $\Delta_k > 0$ and $C_{\max} = \sum_{j=1}^{n} p_j/m$, the job pieces that are processed after a piece of k on any given machine all start later by the same amount. Also, on any given machine, the job pieces that are processed before a piece of job k do not change processing times. Therefore, the starting time of some job $j \neq k$ increases only if it starts after the piece of k on a given machine M_i. If job j is also processed on machine M_{i+1}, it is processed before the piece of k on that machine. Consequently, simultaneous processing of two pieces is only possible for job k. Step 2 of PKADJ prevents this from happening until a total processing time of $m(C_{\max} - p_k)/(m-1)$ is added. Also from Step 2, any remaining processing of job k occurs on machine M_m, when no other job is being processed. Since no job is processed simultaneously on multiple machines, the sequence remains feasible. Because the completion time of each machine increases by the same amount until $p'_k = C_{\max}$, the sequence remains optimal.

When $\Delta_k > 0$ and $C_{\max} > \sum_{j=1}^{n} p_j/m$, the first set of processing time increases does not increase the completion time of any machine. The analysis is now similar to the previous case.

Steps 0, 2 and 4 of WIDE each require $O(n)$ time. Step 1 requires $O(m+n)$ time. Step 3 requires $O(m)$ time. Thus, the overall time complexity of WIDE is $O(m+n)$. $\square$

The schedule found by WIDE is not the only schedule that satisfies the conditions of Theorem 13. Further refinements exist where jobs other than k can have wider sensitivity ranges.

We now establish the companion result that there is no optimal sequence where the processing time of any *arbitrary* job can be made infinitely large and the sequence remains optimal. This negative result is also an example of issue I9.

Theorem 14 *For some $k \in N$, let $p'_k = p_k + \Delta_k$, be a parameter change for $P|pmtn|C_{\max}$. If $n > m$, then there is no sequence that remains optimal for each k and all $\Delta_k > 0$.*

Proof. When Δ_k becomes sufficiently large, $C_{\max} = p'_k$. Once this happens, the schedule remains optimal only if job k is processed throughout the time interval $[0, C_{\max}]$. This implies that job k must start processing at time 0. Because there are only m machines, only m jobs can start at time 0. Consequently, when $n > m$, there is some job k that does not start at time 0. Thus, the sequence becomes nonoptimal for Δ_k sufficiently large. $\square$

It is possible to circumvent our argument in Theorem 14 by including job pieces with zero processing times. However, this is not realistic since scheduling, transporting and setting up a zero processing time job piece in a production shop is costly and confusing to the workforce.

5 Intractable Problems

In this section, we examine sensitivity analysis for intractable problems. First, motivated by issue I10, we explore the relationship between the computational complexity of a problem and the computational complexity of associated sensitivity analysis questions. Then, motivated by issue I11, we suggest approaches to performing heuristic sensitivity analysis for intractable problems.

5.1 Computational Complexity

In general, the computational complexity of performing sensitivity analysis for scheduling problems is not resolved. This is because knowledge of σ^* may affect the work required. The only general result is given by Van Hoesel and Wagelmans (1999) who show that, under reasonable conditions, finding a sensitivity analysis range (question Q1) for an intractable zero-one optimization problem is itself intractable. This implies that answering Q1 for changes in *weights* is intractable, when the scheduling problem is intractable. The next two results show that it may be difficult to develop efficient sensitivity analysis techniques for any intractable scheduling problem. Let $\Psi = \{r_j, p_j, w_j, d_j, \bar{d}_j, prec\}$ be the set of problem class parameters. The following result addresses issue I10.

Theorem 15 *Suppose that problem class P is polynomially solvable. Also suppose that when problem data of type $\omega \in \Psi$ is added to P, the resulting problem class $\hat{P}$ is* NP-hard.

i. Given only the optimal cost, answering Q2 for problem class $\hat{P}$ is NP-hard.

ii. Given the optimal schedule and its cost, there does not exist a polynomial time algorithm that answers Q3 for problem class $\hat{P}$ unless P = NP.

Proof. For part i, consider an arbitrary instance, I, of problem class $\hat{P}$. Remove all data of type ω. Solve this instance of P in polynomial time. Next, change one type ω parameter of one job to match a job in I, and apply a hypothetical polynomial time algorithm to answer Q2. Repeat this procedure for the other jobs. Consequently, the optimal cost for I can be found in polynomial time. This contradicts our assumption about the complexity of $\hat{P}$ and establishes that answering Q2 is binary NP-hard.

The proof is similar for part ii, except that following each parameter change we find in polynomial time both a new optimal schedule and its cost. □

Corollary 2 *Suppose that problem class P is pseudopolynomially solvable. Also, suppose that when problem data of type $\omega \in \Psi$ is added to P, the resulting problem class $\hat{P}$ is unary* NP-*hard.*

i. Given only the optimal cost, answering Q2 for problem class $\hat{P}$ is unary NP-*hard.*

ii. Given the optimal schedule and its cost, there does not exist a pseudopolynomial time algorithm that answers Q3 for problem class $\hat{P}$ unless P = NP.

Proof. Similar to Theorem 15 except that polynomial is replaced by pseudopolynomial. □

The results in Theorem 15 and Corollary 2 address issue I10 for Q2 and Q3.

Remark 1 *Performing sensitivity analysis on the processing times for most* NP-*hard classes of scheduling problems is* NP-*hard.*

When the problem data added is p_j, the instance of P being considered is the one where all processing times are zero. Thus, each job is processed at the earliest possible time. Assuming that a feasible solution exists, it generally can be found in polynomial time. When the constraints create an infeasibility, as in the case when $r_i > \bar{d}_i$ for some $i \in N$ or when the precedence constraints generate a cyclic graph, this also can usually be checked in polynomial time. Exceptions to the remark include scheduling problems where the processing times are specified by the problem definition, such as the unary NP-hard $1|prec, p_j = 1|\sum w_j C_j$ (Lawler, 1978).

5.2 Sensitivity Analysis Using Heuristic Methods

The results in Section 5.1 motivate the use of heuristic sensitivity analysis methods that can estimate the effect of parameter changes in polynomial time. The literature considers Q1 where the original solution is approximate (see for example Chakravarti and Wagelmans, 1998, and Sotskov *et al.*, 1998). There are, however, a variety of other possible heuristic approaches. We consider the issue of how far from the optimal cost a previously optimal schedule can be as a result of a parameter change. This information may make it unnecessary to resolve the problem with the new data if a sufficiently small deviation from optimality is guaranteed.

Two examples are discussed. The first example considers $P2||\sum w_jC_j$. This problem is known to be binary NP-hard (Bruno *et al.*, 1974). The optimization version is solved in pseudopolynomial time by Lawler and Moore (1969). We find bounds for a single parameter change. The value of the bounds and the computational effort depend on the position of the job in σ^*.

Reindex the jobs such that $w_1/p_1 \geq \cdots \geq w_n/p_n$. The following result provides an example of issues I6 and I11 applied to Q2.

Theorem 16 *For some $k \in N$, let $p'_k = p_k + \Delta$ be a parameter change for $P2||\sum w_jC_j$. For σ^*, let B be the set of jobs that are processed after job k and are on the same machine as job k. Let Δ be such that*

a. $w_{k-1}/p_{k-1} \geq w_k/p'_k \geq w_{k+1}/p_{k+1}$.

b. *For job i processed on machine M_1 and job j processed on machine M_2, $C'_i(\sigma^*) \leq C'_j(\sigma^*)$ if and only if $C^*_i \leq C^*_j$, for $1 \leq i \neq j \leq n$.*

Then,

i. *If $\Delta > 0$, then*

$$(z^* + \Delta w_k + \Delta \sum_{i \in B} w_i)/(z^* + \Delta \sum_{i=k}^{n} w_i) \leq z'(\sigma^*)/z'' \leq (z^* + \Delta w_k + \Delta \sum_{i \in B} w_i)/(z^* + \Delta w_k).$$

ii. *If $\Delta < 0$, then*

$$(z^* + \Delta w_k + \Delta \sum_{i \in B} w_i)/(z^* + \Delta w_k) \leq z'(\sigma^*)/z'' \leq (z^* + \Delta w_k + \Delta \sum_{i \in B} w_i)/(z^* + \Delta \sum_{i=k}^{n} w_i).$$

Proof. By construction. $\square$

Observe that $|B|$ is smaller when job k is processed later in the schedule. Consequently, Theorem 16 is a position dependent result in two distinct ways. First, the computational effort required depends on the location of the job in the schedule. The later job k is processed, the less computational work is needed. Second, the value of the bound depends on the location of job k in the schedule. The later job k is processed, the smaller the bound is on $z'(\sigma)/z''$ if $\Delta > 0$ and the greater the bound is if $\Delta < 0$. In the limit, if k is the last job on its machine, then $B = \emptyset$. Thus, $z^* = z''$ (see also Theorem 1) when $\Delta > 0$. Also, if all jobs $i > k$ are in B, then $w_k + \sum_{i \in B} w_i = \sum_{i=k}^{n} w_i$. Thus, $z^* = z''$ when $\Delta < 0$.

We continue our discussion of issue I11 by considering an example of heuristic sensitivity analysis for $P2||C_{\max}$. This problem is binary NP-hard (Karp, 1972). We develop a procedure that finds the interval over which a parameter can change, such that the ratio of the new cost of σ^* to the new optimal cost is bounded above by a specific value. An instance is presented where the bound for Q1 is attainable. We show how to modify σ^* to obtain a heuristic schedule with the same error bound for a larger parameter change. Also, we describe a class of problem instances for which the heuristic procedure finds an exact upper bound for Q1.

We examine the case where the processing time of some job increases. The analysis is similar if the processing time decreases. First, we present the LF (Largest Fit) heuristic that finds a less than exact packing of a subset of items with sizes $a_1, \ldots, a_n$ into a knapsack of size b.

Subroutine LF$(a_1, \ldots, a_n; b)$

```
0.  Let $N = \{1, \ldots, n\}$. Reindex the items such that $a_1 \geq \cdots \geq a_n$.
1.  Find $j = \min\{i \in N \mid a_i < b\}$.
    If $j$ does not exist, then set LF $= b$, which is the unused space, and terminate.
2.  Set $b = b - a_j$ and $N = N \setminus \{j\}$. Go to Step 1.
```

If $\sum_{i=1}^{n} a_i \geq b$, then this subroutine finds a solution with a value that is at least 1/2 of the optimal value. Consider an example where $b = 2$ and where there are three jobs that have sizes $1 + \epsilon, 1$ and 1. For this example, the bound of 1/2 is asymptotically attainable.

Without loss of generality, assume that job n is processed on M_2 and has an increase in processing time of $\Delta > 0$. Let c_i denote the completion time of M_i, for $i = 1, 2$. (The schedule is implied by the context.) Let $\overline{\Delta}$ denote the maximum increase in p_n such that π^* is still optimal. When $\Delta > \overline{\Delta}$, there is a swap of a subset of the jobs on M_1 with a subset of those on M_2, that gives a schedule with smaller makespan. Our heuristic considers some of the possible swaps. If the heuristic finds an improvement, then this establishes that π^* is no longer optimal.

Heuristic SWAP

0. Reindex the machines such that job n is processed on machine M_2 in σ^*. Let A denote the set of jobs that σ^* assigns to machine M_1, and p_A the vector of processing times of these jobs. Let $B = N \setminus (A \cup \{n\})$.
1. Set $\Delta_1 = \text{LF}(p_A; p_i)$, where $i = \text{argmin}_{j \in B}\{p_j\}$.
   Set $\Delta_2 = \text{LF}(p_A; \min_{j \in B \setminus \{i\}}\{p_j\})$.
   Set $\Delta_3 = \text{LF}(p_A; c_2 - p_n)$.
2. $\Delta^u = \min\{\Delta_1, \Delta_2, \Delta_3\} + c_1 - c_2$.

In Step 1, SWAP considers three different packings of the jobs of A into subsets of B. If $\Delta > \Delta^u$, then π^* is no longer optimal. We can swap the selected jobs of A with the corresponding jobs on machine M_2 to reduce the makespan.

Step 0 of SWAP requires $O(n)$ time. The time requirement for Step 1 of SWAP is the time requirement for LF. Step 2 of SWAP requires constant time. Indexing the jobs in Step 0 of LF requires $O(n \log n)$ time. In Step 1 of LF, the selection from an ordered list requires constant time. Since there are at most n selections, the time requirement for LF is $O(n \log n)$. As a result, the overall time complexity of SWAP is $O(n \log n)$.

To establish the worst-case performance of SWAP, several preliminary results are needed. The first result provides values of Δ for which σ^* remains optimal, answering Q1.

Theorem 17 *Let $p'_n = p_n + \Delta$ be a parameter change for $P2||C_{\max}$, where job n is processed on machine M_2. If $0 \leq \Delta \leq c_1 - c_2$, then $z'(\sigma^*) = z''$.*

Proof. By definition of σ', $z'' = z'(\sigma') \leq z'(\sigma^*)$. If $0 \leq \Delta \leq c_1 - c_2$, then $z'(\sigma^*) = z(\sigma^*)$. Thus, $z'(\sigma^*) = z(\sigma^*) \leq z(\sigma') = z'(\sigma')$. Therefore, $z'(\sigma^*) = z'(\sigma')$. □

Theorem 17 implies that if $c_1 > c_2$, then p_n can be increased so that the two machines complete at the same time and the optimal solution does not change.

The next result answers Q1 by finding a case when SWAP provides an exact bound for $\overline{\Delta}$.

Theorem 18 *SWAP finds $\Delta^u = \overline{\Delta}$ for parameter change $p'_n = p_n + \Delta$ when $n \leq 4$.*

Proof. Suppose that M_2 processes only job n. Since $c_2 = p_n$, π^* remains optimal for any $\Delta \geq 0$.

If M_2 processes two jobs, then M_1 processes at most two jobs. Therefore, Δ_1 is the size of the maximal packing of $c_2 - p_n$. Since the only possible swap is to interchange the job in B with those of A, SWAP must find the swap associated with $\overline{\Delta}$.

If M_2 processes three jobs, then M_1 processes only one job. Therefore, Δ_1, Δ_2 and Δ_3 are the sizes of the maximal packing of each job in B and of $c_2 - p_n$. Since these are the only possible swaps, SWAP must find the one associated with $\overline{\Delta}$. □

To answer Q2, we analyze the maximum relative cost error for σ^* that can result from a parameter change. Let $\Delta^s = \min\{\Delta_1, \Delta_2, \Delta_3\}$.

Theorem 19 *For $P2||C_{\max}$, SWAP provides an upper bound for $\overline{\Delta}$ for parameter change $p'_n = p_n + \Delta$ in $O(n \log n)$ time. If $\Delta \leq \Delta^u$, then $z'(\sigma^*)/z'' \leq 8/7$. Further, this bound is attainable.*

Proof. Scale the jobs of each problem instance so that SWAP determines that $\Delta^s = 2$. Although fractional processing times may result, this does not affect our analysis. From Theorem 18, we assume that $n \geq 5$.

Suppose that M_2 processes only job n in σ^*. Job n remains the only job processed by M_2 in an optimal schedule as Δ increases. Thus, π^* remains optimal.

Now, suppose that M_2 processes two jobs in σ^*. SWAP fails to find $\overline{\Delta}$ only when the maximal packing of $c_2 - p_n$ is not found. This can happen only when SWAP selects a larger job from A that is not in the optimal packing. Then, SWAP is unable to select smaller jobs that are part of the optimal packing. There must be at least two smaller jobs in the optimal packing that are not selected by SWAP, because the optimal packing has a total size larger than the one found by SWAP. Since $\Delta^s = \min\{\Delta_1, \Delta_2, \Delta_3\} = 2$ by assumption, each of these smaller jobs has size at least two; otherwise, one of these jobs would be in the packing found by SWAP. Because A contains at least three jobs, each of which has size at least two, $c_1 \geq 6$. Since $\Delta^s = 2$, $z'(\sigma^*) \leq c_1 + 2$. Because the total processing time in the modified problem is at least $2c_1 + 2$, we have that $z'' \geq (2c_1 + 2)/2 = c_1 + 1$. Consequently, $z'(\sigma^*)/z'' \leq (c_1 + 2)/(c_1 + 1) \leq 8/7$.

When M_2 processes three jobs, SWAP finds a packing for B and for each job in B. Thus, SWAP considers the subset of B that is part of an optimal swap. Since SWAP always finds an optimal packing when $|A| \leq 2$ and the packing found by SWAP is suboptimal, it must be the case that $|A| \geq 3$. This implies that $c_1 \geq 6$. Then, using the analysis for the case when M_2 processes two jobs, we conclude that $z'(\sigma^*)/z'' \leq 8/7$.

Finally, suppose that M_2 processes at least four jobs. Since $\Delta^s = 2$, each job in B must have size at least two. Because $|B| \geq 3$, $\sum_{j \in B} p_j \geq 6$. When $c_1 < \sum_{j \in B} p_j$, the makespan can be reduced by moving job n from M_2 to M_1. This implies that σ^* is not optimal. Therefore, $c_1 \geq \sum_{j \in B} p_j \geq 6$. Similar to the case when M_2 processes two jobs, we conclude that $z'(\sigma^*)/z'' \leq 8/7$.

To show that the bound of 8/7 is attainable, consider an instance with six jobs, where the processing times are 3, 3, 2, 2, 2, and 1 for jobs 1 through 6, respectively. In σ^*, jobs 1 and 2 are processed on machine M_1, and jobs 3, 4, 5, and 6 are processed on machine M_2. SWAP gives $\Delta^u = \Delta_1 + c_1 - c_2 = 2 + 6 - 7 = 1$. As a result, $z'(\sigma^*) = 8$. However, $\Delta^u = 1$ implies that $p'_6 = p_6 + \Delta^u = 2$. As a result, σ' has jobs 1, 3 and 4 on machine M_1, and jobs 2, 5 and 6 on machine M_2. Since $z'' = 7$, we have that $z'(\sigma^*)/z'' = 8/7$.

The computational complexity of finding $\bar{\Delta}$ follows from our discussion of SWAP. □

To answer Q2 and Q3, SWAP provides a new heuristic schedule that bounds the optimal cost. Let σ^s be the schedule generated when the swap associated with Δ^s is performed on σ^*.

Theorem 20 *Let $p'_n = p_n + \Delta$ be a parameter change for $P2||C_{\max}$. If $\Delta^u \leq \Delta \leq \Delta^u + 7\Delta^s/3$, then $z'(\sigma^s)/z'' \leq 8/7$. Further, within the interval $[\Delta^u, \Delta^u + 7\Delta^s/3]$, this bound is attainable only for $\Delta \in \{\Delta^u, \Delta^u + 7\Delta^s/3\}$.*

Proof. When $\Delta = \Delta^u$, then $z'(\sigma^s) = z'(\sigma^*)$ by construction. The swap moves a total processing time of Δ^s from M_2 to M_1. Thus, $c_1(\sigma^s) = c_2(\sigma^s) + \Delta^s$ when $\Delta = \Delta^u$. Let z^{LP} denote the optimal cost of the linear programming relaxation for a given Δ. As Δ increases in the interval $[\Delta^u, \Delta^u + \Delta^s]$, $z'(\sigma^s) - z^{LP}$ decreases. When $\Delta = \Delta^u + \Delta^s$, $z'(\sigma^s) - z^{LP} = 0$.

From the proof of Theorem 19, when $\Delta = \Delta^u$, $c'_2(\sigma^*)/c'_1(\sigma^*) = [c_1(\sigma^*) + \Delta^s]/c_1(\sigma^*) \leq 8/6$. This implies that $\Delta^s \leq c_1(\sigma^*)/3$. Therefore, if $\Delta^u \leq \Delta \leq \Delta^u + 7\Delta^s/3$, then

$$z'(\sigma^s)/c_1(\sigma^s) \quad \leq \quad [c_1(\sigma^*) + 7\Delta^s/3] / [c_1(\sigma^*) + \Delta^s]$$

$$\leq \; [c_1(\sigma^*) + 7c_1(\sigma^*)/9]/[c_1(\sigma^*) + c_1(\sigma^*)/3] \; = \; 8/6.$$

Thus, $z'(\sigma^s)/z'' \leq z'(\sigma^s)/z^{LP} \leq z'(\sigma^s)/[(c_1(\sigma^s) + c_2(\sigma^s))/2] \leq 8/7$.

Since $z'(\sigma^s) - z^{LP}$ is strictly decreasing in the interval $[\Delta^u, \Delta^u + \Delta^s]$ and strictly increasing in the interval $[\Delta^u + \Delta^s, \Delta^u + 7\Delta^s/3]$, if $z'(\sigma^s)/z'' = 8/7$, then $\Delta \in \{\Delta^u, \Delta^u + 7\Delta^s/3\}$.

We now show that the bound of 8/7 is attainable for $\Delta \in \{\Delta^u, \Delta^u + 7\Delta^s/3\}$. The example from the proof of Theorem 19 shows that the bound is attainable for $\Delta = \Delta^u$. Consider an instance with six jobs, where the processing times are 10, 8, 6, 6, 6 and 1 for jobs 1 through 6, respectively. In σ^*, jobs 1 and 2 are processed on M_1, and jobs 3, 4, 5 and 6 are processed on M_2. SWAP gives $\Delta^u = 5$ and $\Delta^s = 6$. For $\Delta = \Delta^u + 7\Delta^s/3 = 19$, we have that $p'_6 = 20$. Schedule σ^s processes jobs 1, 2 and 3 on machine M_1 and jobs 4, 5 and 6 on M_2. Hence, $z'(\sigma^s) = 32$. However, σ' processes jobs 1, 3, 4 and 5 on M_1 and jobs 2 and 6 on M_2, and $z'' = 28$. Thus, for $\Delta = \Delta^u + 7\Delta^s/3$, we have that $z'(\sigma^s)/z'' = 8/7$. □

Note that the $O(n)$ time heuristic of He *et al.* (2001) for $P2||C_{\max}$ provides a bound of 12/11. However, this heuristic only holds for a given data set. By contrast, the results in Theorems 19 and 20 apply for all values of Δ within the specified ranges.

6 Concluding Remarks

This paper studies a variety of sensitivity analysis issues. We examine several examples where it is possible to compute a new optimal schedule efficiently by using the existing optimal schedule. Situations are identified where sensitivity analysis results for an optimal schedule can be obtained efficiently by using an optimal sequence. We also present several situations where it is worthwhile to perform additional computations or a different type of computations, to facilitate later sensitivity analysis. In some cases, the results depend on the position of the job with changed parameters in the original optimal sequence. We also discuss the computational complexity of answering sensitivity analysis questions for intractable scheduling problems. For an intractable scheduling problem, we establish worst-case performance bounds on the relative increase above optimal cost when using an original optimal schedule after a parameter change. Finally, we extend the previous analysis to derive heuristic solutions, also with guaranteed worst-case performance, for a parameter change within a larger range.

Our analysis can be applied to numerous situations that are not discussed due to lack of space. We now mention some examples. The methodology of Theorems 4 and 5 can be applied to $1||L_{\max}$, for which Jackson (1955) shows that an earliest due date sequence is optimal. An analysis similar to that in Theorem 3 can be applied to $P2||\sum C_j$. The analysis of release date changes for $1|r_j, pmtn|\sum C_j$ in Theorems 6 and 7 can also be applied to changes in processing times for that problem. The idea of advance planning for sensitivity analysis discussed in Section 4 can also be used for intractable problems. As a final example, the heuristic sensitivity analysis for $P2||C_{\max}$ in Section 5.2 can be applied to similar parallel machine scheduling problems.

We hope that the issues we discuss will provide insights into new sensitivity analysis approaches for other problem classes, including those for which classical sensitivity analysis questions have already been explored.

Acknowledgments

This work is supported in part by the Summer Fellowship Program of the Fisher College of Business, The Ohio State University.

References

Ahuja, R.K., T.L. Magnanti and J.B. Orlin. 1993. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, N.J.

Anderson, D.R., D.J. Sweeney and T.A. Williams. 1999. *Contemporary Management Science With Spreadsheets*. South-Western College Publishing, Cincinnati, OH.

Ball, M.O. and R. Taverna. 1985. Sensitivity Analysis for the Matching Problem and its Use in Solving Matching Problems With a Single Side Constraint. *Annals of Operations Research* **4**, 25–56.

Bazaraa, M.S., J.J. Jarvis and H.D. Sherali. 1990. *Linear Programming and Network Flows*, 2nd edition. John Wiley, New York.

Bradley, S.P., A.C. Hax and T.L. Magnanti. 1977. *Applied Mathematical Programming*. Addison-Wesley, Reading, MA.

Bruno, J.L., E.G. Coffman, Jr. and R. Sethi. 1974. Scheduling Independent Tasks to Reduce Mean Finishing Time. *Communications of the ACM* **17**, 382–387.

Camm, J.D. and T.H. Burwell. 1991. Sensitivity Analysis in Linear Programming Models with Common Inputs. *Decision Sciences* **22**, 512–518.

Chakravarti, N. and A.P.M. Wagelmans. 1998. Calculation of Stability Radii for Combinatorial Optimization Problems. *Operations Research Letters* **23**, 1–7.

Conway, R.W., W.L. Maxwell and L.W. Miller. 1967. *Theory of Scheduling*. Addison-Wesley, Reading, MA.

Cook, W., A.M.H. Gerards, A. Schrijver and E. Tardos. 1986. Sensitivity Theorems in Integer Linear Programming. *Mathematical Programming* **34**, 251–264.

Cozzens, M. and F.S. Roberts. 1991. Greedy Algorithms for T-Colorings of Graphs and the Meaningfulness of Conclusions About Them. *Journal of Combinatorics, Information and Systems Science* **16**, 286–299.

Derigs, U. 1985. Postoptimal Analysis for Matching Problems. *Methods of Operations Research* **49**, 215–221.

Derigs, U. 1988. *Programming in Networks and Graphs*. Springer, New York.

Federgruen, A. and M. Tzur. 1994. Minimal Forecast Horizons and a New Planning Procedure for the General Dynamic Lot Sizing Model: Nervousness Revisited. *Operations Research* **42**, 456–468.

Gabow, H.N. and R.E Tarjan. 1989. Faster Scaling Algorithms for Network Problems. *SIAM Journal on Computing* **18**, 1013–1036.

Gal, T. 1979. *Postoptimal Analyses, Parametric Programming and Related Topics*. McGraw-Hill, Great Britain.

Garey, M.R. and D.S. Johnson. 1979. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Geoffrion, A.M. and R. Nauss. 1977. Parametric and Postoptimality Analysis in Integer Linear Programming. *Management Science* **23**, 453–466.

Graham, R.L. 1975. Bounds on the Performance of Scheduling Algorithms. In *Computer and Job-Shop Scheduling Theory*, ed. E.G. Coffman, Jr., John Wiley, New York.

Graham, R.L., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan. 1979. Optimization and Approximation in Deterministic Machine Scheduling: A Survey. *Annals of Discrete Mathematics* **5**, 287–326.

Greenberg, H.J., 1998. An Annotated Bibliography for Post-Solution Analysis in Mixed Integer Programming and Combinatorial Optimization. In *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*, D.L. Woodruff (ed.), Kluwer Academic Publishers, 97–148.

Hall, N.G. 1986. Single and Multiple Processor Models for Minimizing Completion Time Variance. *Naval Research Logistics Quarterly* **33**, 49–54.

He, Y., H. Kellerer and V. Kotov. 2001. Linear Compound Algorithms for the Partitioning Problem. *Naval Research Logistics*, to appear.

Hillier, F.S. and G.J. Lieberman. 1995. *Introduction to Operations Research*, 6th edition. McGraw-Hill, New York.

Jackson, J.R. 1955. Scheduling a Production Line to Minimize Maximum Tardiness. Research Report 43, Management Science Research Project, University of California, Los Angeles.

James, R.J.W. and J.T. Buchanan. 1996. Robustness of Single Machine Scheduling Problems to Earliness and Tardiness Penalty Errors. Working paper, Department of Management, University of Canterbury, Christchurch, New Zealand.

Kanet, J.J. 1981. Minimizing the Average Deviation of Job Completion Times About a Common Due Date. *Naval Research Logistics Quarterly* **28**, 643–651.

Karp, R.M. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher (eds.), pp. 85-103. Plenum Press, New York.

Karp, R.M. and J.B. Orlin. 1981. Parametric Shortest Path Algorithms With an Application to Cyclic Staffing. *Discrete Applied Mathematics* **3**, 37–45.

Klein, D. and S. Holm. 1979. Integer Programming Post-Optimal Analysis with Cutting Planes. *Management Science* **25**, 64–72.

Kolen, A.W.J., A.H.G. Rinnooy Kan, C.P.M. van Hoesel and A.P.M. Wagelmans. 1994. Sensitivity Analysis of List Scheduling Heuristics. *Discrete Applied Mathematics* **55**, 145–162.

Kravchenko, S.A., Y.N. Sotskov and F. Werner. 1995. Optimal Schedules With Infinitely Large Stability Radius. *Optimization* **33**, 271–280.

Kuhn, H.W. 1955. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly* **2**, 83–97.

Lawler, E.L. 1978. Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints. *Annals of Discrete Mathematics* **2**, 75–90.

Lawler, E.L. and J.M. Moore. 1969. A Functional Equation and its Application to Resource Allocation and Sequencing Problems. *Management Science* **16**, 77–84.

Mahadev, N.V.R., A. Pekeč and F.S. Roberts. 1998. On the Meaningfulness of Optimal Solutions to Scheduling Problems: Can an Optimal Solution be Nonoptimal? *Operations Research* **46**, S120–S134.

Marsten, R.E. and T.L. Morin. 1977. Parametric Integer Programming: The Right Hand Side Case. In *Annals of Discrete Mathematics: Studies in Integer Programming*, P. Hammer et al. (eds.), Elsevier, North Holland.

McNaughton, R. 1959. Scheduling with Deadlines and Loss Functions. *Management Science* **6**, 1–12.

Moore, J.M. 1968. An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs. *Management Science* **15**, 102–109.

Papadimitriou, C.H. and K. Steiglitz. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, N.J.

Piper, C.J. and A.A. Zoltners. 1976. Some Easy Postoptimality Analysis for Zero-One Programming. *Management Science* **22**, 759–765.

Rinnooy Kan, A.H.G. 1976. Machine Scheduling Problems: Classification, Complexity and Computations. Nijhoff, The Hague, Netherlands.

Roberts, F.S. 1990. Meaningfulness of Conclusions from Combinatorial Optimization. *Discrete Applied Mathematics* **29**, 221–241.

Roodman, G.M. 1972. Postoptimality Analysis in Zero-One Programming by Implicit Enumeration. *Naval Research Logistics Quarterly* **19**, 435–447.

Schrage, L.E. 1968. A Proof of the Optimality of the Shortest Remaining Processing Time Discipline. *Operations Research* **16**, 687–690.

Schrage, L.E. and L.A. Wolsey. 1985. Sensitivity Analysis for Branch and Bound Integer Programming. *Operations Research* **33**, 1008–1023.

Smith, W.E. 1956. Various Optimizers for Single Stage Production. *Naval Research Logistics Quarterly* **3**, 59–66.

Sotskov, Y.N. 1991. Stability of an Optimal Schedule. *European Journal of Operational Research* **55**, 91–102.

Sotskov, Y.N., V.K. Leontev and E.N. Gordeev. 1995. Some Concepts of Stability Analysis in Combinatorial Optimization. *Discrete Applied Mathematics* **58**, 169–190.

Sotskov, Y.N., A.P.M. Wagelmans and F. Werner. 1998. On the Calculation of the Stability Radius of an Optimal or an Approximate Schedule. *Annals of Operations Research* **83**, 213–252.

Steele, D.C. 1973. The Nervous MRP System: How to do Battle. *Production and Inventory Management* **16**, 83–89.

Tovey, C.A. 1986. Rescheduling to Minimize Makespan on a Changing Number of Identical Processors. *Naval Research Logistics Quarterly* **33**, 717–724.

Tovey, C.A. 1990. A Simplified Anomaly and Reduction for Precedence Constrained Multiprocessor Scheduling. *SIAM Journal on Discrete Mathematics* **3**, 582–584.

Van Hoesel, C.P.M. and A. Wagelmans. 1999. On the Complexity of Postoptimality Analysis of 0/1 Programs. *Discrete Applied Mathematics* **91**, 251–263.

Wagelmans, A.P.M., 1990. Sensitivity Analysis in Combinatorial Optimization. Ph.D. Thesis, Erasmus University, Rotterdam, Netherlands.

Wagner, H.M. 1995. Global Sensitivity Analysis. *Operations Research* **43**, 948–969.

Weber, G.M. 1981. Sensitivity Analysis of Optimal Matchings. *Networks* **11**, 41–56.

Wendell, R.E. 1985. Tolerance Approach to Sensitivity Analysis in Linear Programming. *Management Science* **31**, 564–578.

Wendell, R.E. 1992. Sensitivity Analysis Revisited and Extended. *Decision Sciences* **23**, 1127–1142.

Wondolowski, F.R., Jr. 1991. A Generalization of Wendell's Tolerance Approach to Sensitivity Analysis in Linear Programming. *Decision Sciences* **22**, 792–810.

Wolsey, L.A. 1981. Integer Programming Duality: Price Functions and Sensitivity Analysis. *Mathematical Programming* **20**, 173–195.