J Sched (2007) 10: 223–235
DOI 10.1007/s10951-007-0014-z

Scheduling for stability in single-machine production systems

Roel Leus · Willy Herroelen

Published online: 11 May 2007

Abstract Robust scheduling aims at the construction of a schedule that is protected against uncertain events. A stable schedule is a robust schedule that changes only little when variations in the input parameters arise. This paper presents a model for single-machine scheduling with stability objective and a common deadline. We propose a branch-and-bound algorithm for solving an approximate formulation of the model. The algorithm is exact when exactly one job is disrupted during schedule execution.

Keywords Single-machine scheduling · Uncertainty · Robustness · Branch-and-bound

1 Introduction

Manufacturing schedules are rarely executed in a 'vacuum' environment and regularly suffer disruptions from a variety of sources such as resource unavailability, tardy deliveries of material or sub-assemblies, altered work content of certain jobs, etc. At the planning stage, uncertainty can be anticipated in multiple ways. A first option is to eliminate the use of schedules altogether and to adhere to a scheduling 'policy', which determines dynamically which jobs to dispatch through time. We refer to Part 2 of Pinedo (2002) for a survey in machine scheduling and to Stork (2001) for a project-scheduling setting.

R. Leus (✉) · W. Herroelen
Department of Decision Sciences and Information Management, Katholieke Universiteit Leuven, Naamsestraat 69, 3000 Leuven, Belgium
e-mail: Roel.Leus@econ.kuleuven.be

W. Herroelen
e-mail: Willy.Herroelen@econ.kuleuven.be

Alternatively, a schedule can be constructed despite the uncertainty inherent in the scheduling environment. Such a *predictive schedule* (*pre-schedule*) or *baseline schedule* has some very important functions. One of these is to allocate resources to the different activities so as to optimize some measure of performance and/or because advance bookings of key staff or equipment are necessary to guarantee their availability. The baseline schedule is also the starting point for communication and coordination with external entities in the company's inbound and outbound supply chain: it constitutes the basis for agreements with suppliers and subcontractors (e.g., for planning external activities such as material procurement and preventive maintenance), as well as for commitments to customers (delivery dates). The usefulness of a predictive schedule is further discussed in Aytug et al. (2005), Mehta and Uzsoy (1998), and Wu et al. (1993).

When disruptions occur during schedule execution, the baseline schedule needs to be rescheduled. If we wish to exploit the coordination purposes of a schedule as well as possible, the actual start of each job should occur as closely as possible to its baseline starting time. We refer to *stability* as a quality of the scheduling environment when there is little deviation between the baseline and the executed schedule. Stability can be aimed for during rescheduling and is then alternatively referred to as *minimally disruptive*, *minimal-perturbation* and *minimum-deviation (re)scheduling*; see for instance Akturk and Gorgulu (1999), Bean et al. (1991), Calhoun et al. (2002), Raheja and Subramaniam (2002), Rangsaritratsamee et al. (2004), and Wu et al. (1993).

A baseline with express anticipation of disruptions, which is protected against certain undesirable consequences of rescheduling, is called *robust*. The option explored in this paper is to introduce stability into the baseline schedule, i.e., as a robustness measure. This stability concept has also been termed *solution robustness* or *predictable scheduling*.

Examples from the literature are sparse; we mention Leus (2003), Mehta and Uzsoy (1998), and O'Donovan et al. (1999).

In the following section we introduce some notation and develop a formal problem statement. Section 3 presents a branch-and-bound algorithm for optimally solving the problem under study, and Sect. 4 discusses a number of benchmark heuristics. Computational experiments with the proposed algorithms are presented in Sect. 5. Finally, conclusions are presented in Sect. 6.

2 Notation and problem statement

2.1 Definitions and objective function

A set of jobs $N = \{1, 2, \ldots, n\}$ with deterministic baseline durations d_i $(i \in N)$ is to be scheduled on a single machine; all jobs are available for processing at the beginning of the planning period. A baseline schedule is an n-vector $\mathbf{s}$, which specifies a starting time s_i for each job i. There is a common deadline ω for all the jobs (e.g., one day's production-shift length): $s_i + d_i \leq \omega, \forall i \in N$. The actual duration of i is a stochastic variable D_i, which need not always equal d_i. The actual starting time $S_i(\mathbf{s})$ of job i is a random variable that is dependent on $\mathbf{s}$ (see Sect. 2.2). Non-negative integer cost c_i is incurred per unit-time deviation in the start time of job i, as a penalty for the resulting system nervousness and shop-coordination difficulties and the delivery delay for the customer. The expected weighted deviation between *actual* and *planned* job starting times is the stability measure for schedule $\mathbf{s}$: we minimise objective function $\sum_{j \in N} c_j |E[S_j(\mathbf{s})] - s_j|$, where $E[\cdot]$ is the expectation operator. In the remainder of the article we omit the argument $\mathbf{s}$ when there is no danger of confusion.

Stochastic job duration D_i is modelled by means of discrete scenarios, a choice that was also made by, e.g., Daniels and Carrillo (1997), Daniels and Kouvelis (1995), Kouvelis and Yu (1997), and Kouvelis et al. (2000). Specifically, let a random variable L_i denote the increase in d_i if i is 'disrupted', which takes place with probability π_i; D_i equals the baseline duration d_i with probability $(1 - \pi_i)$. The variable L_i is discrete with probability-mass function $g_i(\cdot)$, which associates non-zero probability with positive values $l_{ik} \in \Psi_i$, where Ψ_i denotes the set of disruption scenarios for the duration of job i, $\sum_{k \in \Psi_i} g_i(l_{ik}) = 1$ and g_{ik} is used as shorthand for $g_i(l_{ik})$; the l_{ik}-variables are indexed from small to large. Disruption lengths l_{ik} are assumed to be integers, and the D_i for different jobs i are independent. For encoding reasons, we require all values π_i to be rational numbers (represented by two integers), and g_i to map into the set of rational numbers.

2.2 No early start

Stability considerations will often make it undesirable, if not impossible, to commence processing a job earlier than its baseline starting time. We model this restriction by imposing that jobs are not started earlier than planned, i.e., $s_i \leq S_i, \forall i \in N$, which guarantees that actual production will strictly cling to the baseline if no disruptions occur. In effect, the baseline starting times become 'release dates' for schedule execution. This type of constraint is inherent in course scheduling, sports timetabling and railway and airline scheduling. In manufacturing, job execution cannot start before auxiliary resources and tooling have been freed elsewhere in the shop and before the necessary parts and materials have been delivered to the processing site, and the due date communicated to the parties responsible for these prerequisites is normally the baseline starting time at the time of initial schedule development. 'Forbidden early shipment' restrictions at earlier stages in the production process (see, e.g., Christy and Kanet 1990, Kanet and Christy 1984, and Yano 1987) also constitute a source of this behaviour.

When the baseline schedule is implemented, the realization of D_i becomes known when job i is executed. The exact timing of this information is not important since we reschedule by right-shifting the remaining jobs without resequencing. If we define $[i]$ to be the job that is scheduled in the ith position, then

$$\begin{cases} S_{[1]} = s_{[1]}, \\ S_{[i]} = \max\{s_{[i]}; S_{[i-1]} + D_{[i-1]}\}, \quad i = 2, \ldots, n. \end{cases}$$

The 'no early start' assumption has a major impact on the model. Removing this assumption, however, leaves two options:

1. All jobs are executed contiguously from time 0. This does not seem useful considering the particular choice for the objective function.
2. Additional decisions need to be taken *during* schedule execution in order to determine when exactly each job is to be started. The additional complexity of this choice appears to be virtually impossible to deal with.

For reasons of tractability, we will, therefore, examine only the situation where the 'no early start' restriction is imposed.

2.3 Model formulation

The problem under study has an irregular objective function: an optimal solution need not necessarily exist without inserted idle time and a permutation of the jobs may not suffice to produce a solution. A survey of classical scheduling problems with this characteristic is given in Kanet and Sridharan (2000). In our particular environment of variable job durations, inserted idle time can be envisaged as buffer time,

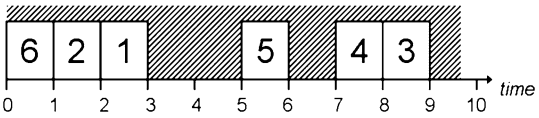

Fig. 1 Schedule for the example problem when $\omega = 9$.

used to cushion the propagation of a disruption towards the successors of the disrupted job. We illustrate this problem setting by means of a brief example. Consider a problem instance with $n = 6$ jobs where all jobs have equal duration $d_i = 1$, and a time horizon of $\omega = 9$ time units is allotted to the set of jobs. Consequently, we have three spare units of time that can serve as a buffer. A feasible solution to this instance is depicted in Fig. 1. In this schedule the starting time of job 5 is protected from a disruption of up to two time units in the duration of jobs 1, 2 or 6.

The following decision variables are defined:

$$X_{ip} = \begin{cases} 1, & \text{if job } i \text{ is processed in position } p, \\ 0, & \text{otherwise,} \end{cases}$$
$$i, p = 1, \ldots, n.$$

Values X_{ip} are gathered into the vector $\mathbf{x}$. $\mathcal{X}$ represents the set of 0/1-vectors that satisfy (1) and (2), which ensure that each position corresponds with exactly one job:

$$\sum_{p=1}^{n} X_{ip} = 1, \quad i = 1, \ldots, n, \tag{1}$$

$$\sum_{i=1}^{n} X_{ip} = 1, \quad p = 1, \ldots, n. \tag{2}$$

There is a one-to-one correspondence between each $\mathbf{x} \in \mathcal{X}$ and a total order[1] $T(\mathbf{x})$ of N: for any $\mathbf{x} \in \mathcal{X}$,

$$T(\mathbf{x}) = \big\{(i, j) \subset N \times N \mid \exists p, q \in \{1, \ldots, n\} : p < q \wedge X_{ip} X_{jq} = 1\big\}.$$

A second set of decision variables is the collection of

$F_p =$ the size of the buffer immediately after the job in position p $(p = 1, \ldots, n)$.

Values F_p are collected into the vector $\mathbf{f}$. $\mathcal{F}(\omega)$ is the set of (component-wise) non-negative vectors $\mathbf{f}$ that comply with the following equation (specifying the available total buffer space):

$$\sum_{p=1}^{n} F_p = \omega - \sum_{i=1}^{n} d_i.$$

A set of sequencing decisions $\mathbf{x}$ and buffer sizes $\mathbf{f}$ completely determines a baseline schedule $\mathbf{s}(\mathbf{x}, \mathbf{f})$ in the following way:

$$s_i(\mathbf{x}, \mathbf{f}) = \sum_{p=1}^{n} X_{ip} \left(\sum_{q=1}^{p-1} \left(F_q + \sum_{j=1}^{n} X_{jq} d_j \right) \right), \quad i = 1, \ldots, n. \tag{3}$$

Note that implicitly $s_{[1]} = 0$.

This enables us to provide a conceptual formulation for the problem under study, subsequently referred to as STABILITY:

$$\min \quad \sum_{i=1}^{n} c_i \big(E[S_i] - s_i\big), \tag{4}$$

subject to

$$S_{[1]} = s_{[1]}(\mathbf{x}, \mathbf{f}),$$
$$S_{[i]} = \max\{s_{[i]}(\mathbf{x}, \mathbf{f}); S_{[i-1]} + D_{[i-1]}\}, \quad i = 2, \ldots, n,$$
$$\mathbf{x} \in \mathcal{X}, \qquad \mathbf{f} \in \mathcal{F}(\omega).$$

In what follows, the arguments to $\mathbf{s}$ are not mentioned if there is no risk of confusion.

To evaluate the objective-function value (4) for a feasible solution $\mathbf{s}$, little less seems to be possible than to evaluate all $\prod_{i \in N} (|\Psi_i| + 1)$ possible combinations of duration disruptions. In line with Elmaghraby (2005), we observe that "any approach that aspires to confront uncertainty head-on is computationally overwhelming". Evaluation can be performed in pseudo-polynomial time $O(n^2 l_{\max} \Psi_{\max})$, with $\Psi_{\max} = \max_{i \in N} |\Psi_i|$ and $l_{\max} = \max_{i \in N} l_{i|\Psi_i|}$, similar to the 'forward-backward' algorithm for the determination of the distribution of a sum of independent discrete random variables (see, for instance, Fearnhead and Meligkotsidou 2004). As this remains computationally unattractive, we develop a model that focuses only on the main effects of the separate disruption of each of the n jobs rather than on all possible disruption interactions.

Define I_i to be the indicator variable that is 1 if job i is disrupted, and 0 otherwise, so $K := \sum_{i \in N} I_i$ is the number of disrupted activities. The objective function (4) is altered as follows, yielding problem STABILITY WITH ONE DISRUPTION (SWOD):

$$\min \quad \sum_{i=1}^{n} c_i \big(E[S_i | K = 1] - s_i\big). \tag{5}$$

[1] An order relation is a subset of the Cartesian product $C \times C$ of its ground set C (in the context of the paper, a set of job pairs) fulfilling the requirements of irreflexivity, anti-symmetry and transitivity. A *complete* or *total* order relation R on C additionally satisfies the comparability condition that either $(a, b) \in R$ or $(b, a) \in R$ for any $a, b \in C, a \neq b$.

The model assumes that exactly one job suffers a disruption from its baseline duration. The resulting restricted model is useful when disruptions are sparse and spread over time, so that the number of interactions is limited. Our computational results (Sect. 5) show that the model is quite robust to variations in the expected number of disrupted jobs $\sum_{i\in N}\pi_i$. We elaborate on the validity of the model in Sect. 2.5. Stand-alone evaluation of this objective function requires $O(n^2\Psi_{\max})$ time. Comparable restrictions of scope have been made by Adiri et al. (1989) (a single deterministic or stochastic machine breakdown), Leon et al. (1994) (one disruption on a single fallible machine in a job shop), and Mehta and Uzsoy (1998) (the distance from a schedule with all jobs disrupted is minimized). Finally, a reasoning quite akin to ours in a graph-coloring context can be found in Yáñez and Ramírez (2003), where the robustness of a coloring is measured as the probability of the coloring remaining valid after *one* random complementary edge is added to the edge set.

The following probabilities are readily computed:

$$P[0] := P[K=0] = \prod_{i=1}^{n}(1-\pi_i),$$

$$P[1] := P[K=1] = \sum_{i=1}^{n}\pi_i \prod_{\substack{j=1\\ j\neq i}}^{n}(1-\pi_j),$$

and more generally

$$P[k] := P[K=k] = \sum_{\substack{V\subseteq N\\ |V|=k}}\left(\prod_{i\in V}\pi_i\right)\left(\prod_{i\in N\setminus V}(1-\pi_i)\right),$$

$$k=0,1,\ldots,n,$$

assuming that $\prod_{i\in\varnothing}(\cdot)=1$. We can also compute the values

$$p_i := P[I_i=1|K=1] = \left(\pi_i\prod_{j\neq i}(1-\pi_j)\right)\Big/P[1],$$

$$i=1,\ldots,n, \tag{6}$$

representing the probability that job i is the unique disrupted job, conditional on exactly one job being disrupted.

We define an additional decision variable:

$\Delta_{ijk} =$ the delay in the start time of job j due to a disruption according to scenario k of job i, when $K=1$.

SWOD can now be formulated as follows, with $\alpha_{ijk} = p_i g_{ik} c_j$:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{|\Psi_i|}\alpha_{ijk}\Delta_{ijk}, \tag{7}$$

subject to

$$\Delta_{ijk} + \sum_{r=p}^{q-1} F_r \geq l_{ik}(X_{ip}+X_{jq}-1),$$

$$i,j,p,q=1,\ldots,n;\ i\neq j;$$

$$k=1,\ldots,|\Psi_i|;\ p<q, \tag{8}$$

$$\text{all } \Delta_{ijk}\geq 0, \tag{9}$$

$$\mathbf{x}\in\mathcal{X}, \qquad \mathbf{f}\in\mathcal{F}(\omega). \tag{10}$$

In the objective (7) the expected value of the starting-time delay of job j is computed by summing the values Δ_{ijk} weighted with probability $p_i g_{ik}$ and cost c_j. Restrictions on the values Δ_{ijk} are imposed by (8) for indexes i and j that are assigned to positions p and q, respectively (the other equations are not restrictive). The corresponding delay in the start time of job j due to disruption in job i will be equal to zero or $l_{ik} - \sum_{r=p}^{q-1} F_r$, the disruption length of i minus the buffer size in place between the positions p and q, whichever is larger.

2.4 Illustration

We continue the example introduced in Sect. 2.3. Tasks indexed 5 and 6 are considered to be of high importance, the cost of delay in their starting times is $c_5 = c_6 = 4$; the other jobs $i\neq 5,6$ have $c_i = 1$. Remaining data is provided in Table 1. Job 1, for instance, has a probability of three out ten of suffering a duration disruption, and, if this occurs, duration will increase by either one or two time units, both cases equally likely.

An optimal solution to the corresponding instance of SWOD is depicted in Fig. 1; the optimal objective-function value is 0.804962. Clearly, the available idle time is put to good use: if we reduce ω to 6 (no idle time anymore), the optimal solution attains an associated cost of 3.45742 for job sequence 6–2–5–4–1–3. Sequence 6–2–1–5–4–3 (optimal for $\omega = 9$) corresponds with a cost of 5.08227 when $\omega = 6$, whereas 6–2–5–4–1–3 achieves a cost of at least 1.25092 when the scheduling horizon is nine time units.

2.5 Properties

The scheduling problem SWOD as set out above has been shown to be $\mathcal{NP}$-hard in the ordinary sense by Leus and Herroelen (2005), even if all $|\Psi_i| = 1$, assuming that all p_i are rational numbers. This study used a reduction from $P2||\sum w_j C_j$ (whose decision-problem version was proved to be ordinarily $\mathcal{NP}$-complete via reduction from SUBSET SUM by Bruno et al. 1974). A similar proof can be set up to show strong $\mathcal{NP}$-hardness by reduction from $P||\sum w_j C_j$,

Table 1 Disruption data for the example problem. Values p_i are computed according to (6). $P[0] = 0.282791$ and $P[1] = 0.414383$, so $\sum_{k=2}^{n} P[k] = 0.302826$

Job i	1	2	3	4	5	6
π_i	0.3	0.05	0.3	0.1	0.25	0.1
$\|\Psi_i\|$	2	2	1	2	2	1
l_{i1} (g_{i1})	1 (0.5)	1 (0.7)	2 (1)	2 (0.5)	1 (0.5)	2 (1)
l_{i2} (g_{i2})	2 (0.5)	2 (0.3)	–	4 (0.5)	2 (0.5)	–
p_i	0.292474	0.035918	0.292474	0.075827	0.22748	0.075827
$\pi_i E_i[L_i]/c_i$	0.45	0.065	0.6	0.3	0.09375	0.05
$p_i E_i[L_i]/c_i$	0.438712	0.046693	0.584949	0.22748	0.085305	0.037913

which is said by the website[2] on complexity results for scheduling problems maintained at the University of Osnabrück to have been shown strongly $\mathcal{NP}$-hard, based on an unpublished reference by Lenstra.

We also show the following result, which is reassuring in view of our difficulties when dealing with STABILITY:

Theorem 1 *Problem* STABILITY *is $\mathcal{NP}$-hard.*

For the proof of Theorem 1, we first derive some intermediate results.

Lemma 1 *For any given set of values $p_i \in [0; 1], i \in N$, there exists a corresponding set of values $\pi_i \in [0; 1], i \in N$, fulfilling the set of equations (6).*

Proof From (6) we derive

$$p_i P[1](1 - \pi_i) = \pi_i P[0] \quad \forall i \in N$$

or

$$\left(\frac{P[1]}{P[0]}\right) p_i = \frac{\pi_i}{1 - \pi_i} \quad \forall i \in N. \tag{11}$$

An arbitrary choice of $\pi_i \in [0; 1]$ for one job i determines $(\frac{P[1]}{P[0]})$ and, thereby, the other π-values. All $\pi_i \in [0; 1]$ because all $\pi_i/(1 - \pi_i) \geq 0$. □

Define $\pi_{\max} = \max_{i \in N} \pi_i$.

Lemma 2 *For sufficiently low $\pi_{\max}$, each optimal schedule to* STABILITY *is also optimal to the corresponding instance of* SWOD *it is derived from by means of (11).*

Proof The objective function for STABILITY is

$$\sum_{i=1}^{n} c_i E[S_i - s_i | K = 1] P[1] + \sum_{i=1}^{n} c_i \sum_{k=2}^{n} E[S_i - s_i | K = k] P[k]. \tag{12}$$

The absolute value of the smallest possible non-zero change in the first term of (12) is $\geq \frac{P[1]}{G}$, with G the product of the denominators of the values α_{ijk} (as an upper bound on their least common multiple). In SWOD it is not difficult to show that we can restrict attention to integer values for Δ_{ijk}. Therefore, it suffices to show that, for suitably selected values π, $\frac{P[1]}{G}$ exceeds (in absolute value) the largest possible change in the second term of (12), so that any optimal solution to the STABILITY-instance automatically optimizes the SWOD-objective (5), or

$$\frac{P[1]}{G} > \sum_{i=1}^{n} c_i \sum_{k=2}^{n} E[S_i - s_i | K = k] P[k]. \tag{13}$$

The right-hand side of (13) is smaller than or equal to

$$n^2 c_{\max} l_{\max} \sum_{k=2}^{n} P[k],$$

and

$$\sum_{k=2}^{n} P[k] \leq \sum_{k=2}^{n} \binom{n}{k} \pi_{\max}^2 < 2^n \pi_{\max}^2,$$

with $c_{\max} = \max_{i \in N} c_i$. $P[1]$ in turn is $\geq \pi_{\max}(1 - \pi_{\max})^{n-1}$. Hence, (13), and a fortiori (12), certainly holds if

$$\frac{\pi_{\max}(1 - \pi_{\max})^{n-1}}{G} \geq n^2 c_{\max} l_{\max} 2^n \pi_{\max}^2$$

or

$$\frac{(1 - \pi_{\max})^{n-1}}{\pi_{\max}} \geq C,$$

[2] http://www.mathematik.uni-osnabrueck.de/research/OR/class/.

with $C = n^2 c_{\max} l_{\max} 2^n G$. An equivalent condition is

$$1 \geq \sqrt[n-1]{C\pi_{\max}} + \pi_{\max}, \tag{14}$$

which certainly holds if $\pi_{\max} \leq (2^{n-1}C)^{-1}$, in which case each of the two terms on the right of the latter inequality is ≤ 0.5. □

Equality in (14) leads to $\pi_{\max} = (C+1)^{-1}$ for $n = 2$, but a solution for general n does not seem to be straightforward, which is the reason why we (further) underestimate the required $\pi_{\max}$ for a *sufficient* condition. Lemmas 1 and 2 allow for a polynomial reduction from SWOD to STABILITY, which proves Theorem 1: $\pi_{\max}$ corresponds to each activity with maximal p_i and can, therefore, be appropriately chosen (in polynomial time). At the same time this substantiates our earlier claim that SWOD produces high-quality schedules in case of suitably scarce disruptions. Nevertheless, not all optima for SWOD are valid for STABILITY. In the example discussed in Sect. 2.4 and in Fig. 1, $E[S_5|K=1] = s_5$, and so increasing c_5 does not change optimality for SWOD of this schedule. For STABILITY, however, large enough c_5 will lead to a lower objective-function value when $s_5 = 6$.

When the available float is zero, i.e., for the case $\omega = \sum_{i \in N} d_i$ (= 6 for the example), ordering the jobs in non-decreasing expected weighted disruption length $p_i E_i[L_i]/c_i$, with $E_i[\cdot]$ the expectation operator with respect to L_i, leads to an optimal schedule to SWOD, which is easily shown by an adjacent-interchange argument. The same holds for STABILITY for quantity $\pi_i E_i[L_i]/c_i$. We refer to this rule as the *EWDL*-rule (for *expected weighted disruption length*); the rule always leads to the same sequence(s) for the two problems. Application to the example problem leads to the sequence 6–2–5–4–1–3.

Protection of the deadline is not taken up separately in this paper, since this can be modelled by introducing a dummy job with sufficiently large expected disruption length, so that the dummy is scheduled last in any optimal schedule. Finally, we also have the following result:

Lemma 3 *Without loss of generality, we can set all job durations equal to zero, if we accordingly subtract* $\sum_{i=1}^{n} d_i$ *from* ω.

Proof Model (7–10) remains unchanged if the proposed change is made. □

Based on this lemma, all durations are assumed to be zero in the remainder of this paper.

3 An implicit-enumeration algorithm

In light of the discussion in the previous section on the complexity status of SWOD, an exact algorithm with better than exponential time complexity is unlikely to exist, which is why we devise a branch-and-bound algorithm to perform implicit enumeration of the solution space.

3.1 General approach of the branch-and-bound algorithm

We develop a branch-and-bound (B&B) algorithm for solving SWOD. From front to back of the machine, we fill one job position at each level of the search tree. In this way we gradually partition $\mathcal{X}$ into subsets $\mathcal{X}_h$, which are defined by order relations A_h on N: $\mathcal{X}_h = \{\mathbf{x} \in \mathcal{X} : A_h \subseteq T(\mathbf{x})\}$. The subscript h represents the index of the corresponding search-tree node. $\varphi(N, A_h, \omega)$ denotes the best (minimal) objective-function value reachable by any individual $\mathbf{x} \in \mathcal{X}_h$ (with deadline ω). We initialize activity set $J_0 = \varnothing$ and order relation $A_0 = \varnothing$. Branching from node l to node h corresponds to the selection of one element $\sigma_l \in N \backslash J_h$, and we construct $J_l = J_h \cup \{\sigma_l\}$ and $A_l = A_h \cup \{(i, \sigma_l) \mid i \in J_h\}$. A_h is a complete order on J_h. When $J_h = N$, $\mathcal{X}_h$ is a singleton and the restricted problem boils down to inserting buffers into a fully specified job sequence.

An illustration of the branching scheme is provided in Fig. 2. Nodes in the search tree are numbered in order of exploration and we traverse the tree in a *depth-first* manner (or *last-in-first-out*), since at low-indexed levels the bounds are not very tight, and because we can reduce the computations in a node by using information from its direct parent node (see Sect. 3.4). The latter benefit has been referred to as the *calculation-restart* advantage (Parker and Rardin 1988).

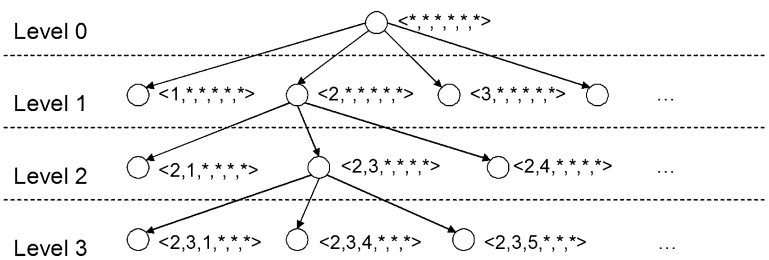

Fig. 2 Illustration of the branching scheme. Set A_h is described next to each node h

3.2 Bounding the objective function

By re-arranging and simplifying the terms in formulation (7–10) (with zero durations, cf. Lemma 3), we derive the following formulation for SWOD in search node h.

$$\varphi(N, A_h, \omega) = \min \sum_{(i,j)\in T(\mathbf{x})} \sum_{k=1}^{|\Psi_i|} \alpha_{ijk} \Delta_{ijk}, \tag{15}$$

subject to

$$s_i(\mathbf{x}) \le s_j(\mathbf{x}), \quad (i,j) \in T(\mathbf{x}), \tag{16}$$

$$s_i(\mathbf{x}) \le \omega, \quad i \in N, \tag{17}$$

$$s_i(\mathbf{x}) + l_{ik} - s_j(\mathbf{x}) \le \Delta_{ijk}, \quad (i,j) \in T(\mathbf{x}),\ k \in \Psi_i, \tag{18}$$

$$\text{all } \Delta_{ijk}, \quad s_i(\mathbf{x}) \ge 0, \tag{19}$$

$$\mathbf{x} \in \mathcal{X}_h. \tag{20}$$

There is a one-to-one correspondence between a set of non-negative starting times fulfilling (16) and (17), on the one hand, and a set of feasible buffer sizes, on the other hand; the constraint $\mathbf{f} \in \mathcal{F}(\omega)$ is, therefore, redundant and not retained (in notations $\mathbf{f}$ is also eliminated as an argument to $\mathbf{s}$). Equations (18) determine the disruption lengths and are obtained from (8): first we add $\sum_{r=1}^{p-1} F_r - \sum_{r=1}^{p-1} F_r\ (=0)$ to the left-hand side of (8). Next we eliminate all X-values by retaining only the necessary running indexes by means of $T(\mathbf{x})$; the starting times s_i and s_j, as defined in (3), can then be recognized.

For arbitrary h, let Δ^*_{ijk} be the set of Δ_{ijk}-values in an optimum for model (15–20). In any search node h, it holds that

$$\sum_{i\in J_h} \sum_{j\in N\setminus J_h} \sum_{k=1}^{|\Psi_i|} \alpha_{ijk} \Delta^*_{ijk}$$

$$\ge \sum_{i\in J_h} \sum_{j\in N\setminus J_h} \sum_{k=1}^{|\Psi_i|} \alpha_{ijk} \max\{0; l_{ik} - \omega\} = q(\omega, h),$$

because $N\setminus J_h$ is appended after the chain of jobs J_h, and no more than ω time units can be inserted between i and j.

Every feasible solution assigns float quantity f $(0 \le f \le \omega)$ to $N\setminus J_h$ (to be inserted between $N\setminus J_h$-jobs) and $(\omega - f)$ is available for J_h, if we neglect the buffer $F_{[|J_h|]}$. Therefore, in any search node h,

$$\varphi(N, A_h, \omega) \ge q(\omega, h) + \min_{0\le f\le\omega} \big\{\varphi(J_h, A_h, \omega - f) + \varphi(N\setminus J_h, \varnothing, f)\big\}. \tag{21}$$

In Sect. 3.4 we discuss how function $\varphi(J_h, A_h, \cdot)$ is actually computed. $\varphi(N\setminus J_h, \varnothing, f)$ is further bounded from below in two different ways. The first bound lb_1 exploits the fact that scheduling with zero float is easy (we use the *EWDL*-rule). We create an auxiliary problem with job set $N\setminus J_h$, in which the disruption length in each scenario k of each job i is set equal to $\max\{0; l_{ik} - f\}$, and the deadline is 0. lb_1 does indeed constitute a lower bound because the available float is re-used between each job pair. Plugging lb_1 into (21) yields the lower bound LB_1 for $\varphi(N, A_h, \omega)$.

A different bound lb_2 for $\varphi(N\setminus J_h, \varnothing, f)$ is based on Jensen's Inequality: if we replace all disruption scenarios $k \in \Psi_i$ of the jobs $i \in N\setminus J_h$ by one single disruption with length $E_i[L_i]$, the resulting objective function is a lower bound to that of the original problem. Replacing all cost coefficients c_i by c_{i^*} with i^* being the job in $N\setminus J_h$ with the lowest cost, and, likewise, taking the same lowest probability and disruption length for all jobs does not increase the objective value. For the resulting set of $|N\setminus J_h|$ *identical* jobs sequencing is no longer needed and optimal starting times can be obtained by means of network-flow techniques (see Sect. 3.4). We call the resulting bound LB_2.

Unfortunately, the determination of both LB_1 and LB_2 requires a significant amount of computational effort. Both lb_2 and φ are convex in f enabling Golden Section Search; LB_1 is computed by considering all discrete values for $f \in [0; \omega]$ – cf. also the Appendix. We have, therefore, also implemented 'simpler' lower bounds $SLB_x = q(\omega, h) + \varphi(J_h, A_h, \omega) + lb_x(\omega, h)$, $x = 1, 2$, in which both terms in the expression to be minimized in (21) receive the maximum float ω. The SLB_x-bounds are never tighter than their LB_x-counterparts due to the monotonicity of φ and lb_x in f. Preliminary computational experience has indicated that the use of SLB_1 and SLB_2 leads to a more efficient overall algorithm, which is why we restrict ourselves to this choice in the experiments of Sect. 5.

3.3 Further algorithmic details

Dominance rule 1 A pairwise-interchange argument shows that for any two *consecutive* jobs i, j in an optimal solution either $p_i c_j E_{L_j} L_j \le p_j c_i E_{L_i} L_i$ or a non-zero buffer should be inserted between the two positions; otherwise, the solution is dominated. We restrict our search to integral buffer sizes, since an optimal solution exists with integral starting times, which follows from our discussion in Sect. 3.4. Hence, 'non-zero' leads to '$\ge$1', and this additional constraint is explicitly imposed on the starting times of the jobs in J_h. When the cumulative minimal buffer sizes exceed ω, the current search node can be fathomed; this test is performed implicitly by the flow computations in Sect. 3.4. In any node h of the search tree, we let δ^h_{ij} denote the minimal distance between i and j. This gives rise to a starting time constraint in the form of (22) to replace (16):

$$s_i(\mathbf{x}) + \delta^h_{ij} \le s_j(\mathbf{x}), \quad (i,j) \in T(\mathbf{x}). \tag{22}$$

The cumulative δ-values can be subtracted from the float f that is available for jobs $N\backslash J_h$ in lower-bound computations. The impact of dominance rule 1 is explained in the computational experiments.

Dominance rule 2 Jobs with zero cost coefficient can be sequenced last: this is always a dominant decision. Similarly, jobs i with zero $p_i E_{L_i} L_i$ can be scheduled first on the machine without loss of better solutions. Dominance rule 2 has less impact than rule 1 but does not consume a significant amount of CPU-time either, so its impact is not discussed in Sect. 5.

Order of exploration Due to the fact that the lower-bound computations are intimately tied to the incremental construction of solutions (see Sect. 3.4), it is difficult to use them as the basis for determining the order of exploration of the child nodes of a node in the search tree, since the bounds would then need to be computed for *all* branching alternatives before one of the alternatives is implemented. We, therefore, order the candidate jobs in decreasing order of a *pseudo-cost* of insertion, which is an estimate of the true cost but not a bound. The role of this pseudo-cost is in guiding heuristic decisions in the algorithm, not in generating incumbent solutions or in proving fathomability (Parker and Rardin 1988). In our implementation we simply scan the branching alternatives in *EWDL*-order.

3.4 Network flows

In the single-disruption setting outlined in the previous sections, Herroelen and Leus (2004) have examined how to schedule activities without resource constraints but subject to a partial order. This solution method is invoked to compute $\varphi(J_h, A_h, \omega)$. J_h is augmented with jobs 0 and $(n+1)$, both with zero cost and zero disruption probability, which come first and last in A_h, respectively. $\delta^h_{ij} = 0$ if i or j are 0 or $(n+1)$. We obtain the formulation below. The model focuses on the relative position of the jobs in time rather than on absolute values of starting times, which is reflected in the absence of sign constraints for the s-variables:

$$\min \quad \sum_{(i,j)\in A_h} \sum_{k=1}^{|\Psi_i|} \alpha_{ijk} \Delta_{ijk}, \tag{23}$$

subject to

$$s_j - s_i \geq \delta^h_{ij}, \quad (i,j) \in A_h, \tag{24}$$

$$\Delta_{ijk} + s_j - s_i \geq l_{ik}, \quad (i,j) \in A_h,\ k \in \Psi_i, \tag{25}$$

$$s_0 - s_{n+1} \geq -\omega, \tag{26}$$

$$\text{all } \Delta_{ijk} \geq 0; \qquad \text{all } s_i \text{ unrestricted in sign.} \tag{27}$$

If we assign non-negative multipliers x_{ij}, y_{ijk} and v to the constraints (24), (25) and (26), respectively, the dual of the foregoing linear program can be written as follows:

$$\max \quad \sum_{(i,j)\in A_h} \delta^h_{ij} x_{ij} + \sum_{\substack{(i,j)\in A_h \\ k\in\Psi_i}} l_{ik} y_{ijk} - \omega v, \tag{28}$$

subject to

$$\sum_{(i,j)\in A_h} x_{ij} - \sum_{(j,i)\in A_h} x_{ji} + \sum_{\substack{(i,j)\in A_h \\ k\in\Psi_i}} y_{ijk} - \sum_{\substack{(j,i)\in A_h \\ k\in\Psi_j}} y_{jik}$$

$$= \begin{cases} 0, & i \in J_h, i \neq 0, n+1, \\ v, & i = 0, \\ -v, & i = n+1, \end{cases} \tag{29}$$

$$0 \leq y_{ijk} \leq \alpha_{ijk}; \qquad 0 \leq x_{ij}, \quad (i,j) \in A_h,\ k \in \Psi_i. \tag{30}$$

This is a minimum-cost network-flow problem (MCNFP) with the node set J_h and the arc set A_h together with the return arc $(n+1, 0)$. Each arc $(i, j) \in A_h$ is actually a multi-arc, representing $|\Psi_i|+1$ individual arcs with flow quantities x_{ij} and y_{ij1} to $y_{ij|\Psi_i|}$; x_{ij} has the lowest profit $\delta^h_{ij} = 0$ or 1 and is incapacitated, while y_{ijk} has profit coefficient l_{ik} and flow capacity α_{ijk}.

Each MCNFP is solved using an implementation of the strongly polynomial minimum-mean cycle-canceling algorithm (Ahuja et al. 1993), in which the successive negative-cost augmenting directed cycles in the residual network are identified by the algorithm of Karp (1978) as the negative cycles with minimum mean cost (the mean cost of a cycle is its cost divided by the number of arcs it contains). Note that the residual network is always strongly connected because of the presence of the incapacitated x-arcs and return flow v, so that Karp's algorithm is easily implemented. Efficiency enhancements such as those proposed by Dasdan and Gupta (1998) have been tested but are of little value because of the density of the network.

An optimal solution to model (28–30) for search node k constitutes a good feasible starting solution for the new search nodes branched into from k. In order to maintain a feasible flow on backtracking, the flow on arcs whose capacity is re-set to zero is re-routed to the x-arcs.

If the MCNFP is unbounded, the primal model is infeasible because the cumulative minimal starting-time differences δ^h_{ij} exceed ω, in which case we fathom the current search node and backtrack. Otherwise, once an optimal MCNFP-solution is found, an optimal solution to model (23–27) can be constructed via complementary slackness; this is only necessary for storing new incumbents at level n. The following cases are distinguished:

1. $y_{ijk} = 0$. Since $\Delta_{ijk} = 0$ (complementary slackness), $s_j \geq s_i + l_{ik}$.

2. $0 < y_{ijk} < \alpha_{ijk}$. This leads to $\Delta_{ijk} + s_j - s_i = l_{ik}$ (complementary slackness) and again $\Delta_{ijk} = 0$, so $s_j = s_i + l_{ik}$.
3. $y_{ijk} = \alpha_{ijk}$. In this case, $\Delta_{ijk} + s_j - s_i = l_{ik}$, and since $\Delta_{ijk} \geq 0$, we obtain that $s_i \geq s_j - l_{ik}$.

For the x-arcs, we have

1. $x_{ij} = 0$. This gives $s_j \geq s_i + \delta_{ij}^h$.
2. $x_{ij} > 0$. This yields $s_j = s_i + \delta_{ij}^h$.

Using these observations we find a solution to the primal problem by solving a longest-path problem in the residual network (which may have negative arc lengths), where arc lengths are equal to the minimum timelags between the job starting times. The longest path from 0 to i minimizes s_i subject to the equality and inequality constraints. Without loss of better solutions, we choose $s_0 = 0$ and $s_{n+1} = \omega$. The remaining starting times are well-defined because the residual network does not contain a positive cycle, from optimality of the MCNFP-solution. Because of the structure of the profit coefficients, at most one arc corresponding to each multi-arc carries the flow at a value strictly between its lower and upper bounds, which allows us to identify the predecessor disruption scenario up to which jobs are protected. The longest-path problem is solved using an adaptation of the FIFO label-correcting algorithm: from s_0 and s_{n+1}, we can obtain permanent starting times for intermediary jobs i if equality restrictions relate s_i to other permanent starting times (while in principle, for label-correcting algorithms such as the FIFO algorithm, all labels are temporary until termination of the algorithm, see Ahuja et al. 1993).

4 Heuristics

The performance of the proposed one-disruption model for the stability objective is compared with four simple heuristics. Two possibilities are considered for sequencing: a full order on N is determined (1) as the *EWDL*-order ("*E*") and (2) randomly (in increasing order of job index, "*I*"). Note that *any* job sequence leads to feasible primal solutions.

After the sequencing phase, the jobs are scheduled (or buffers inserted), (1) optimally, using the network-flow techniques of Sect. 3.4 ("*N*"), and (2) by means of the *ADFF*-heuristic ("*A*"). *ADFF* (*activity-dependent float factor*), proposed in Herroelen and Leus (2004) in a slightly adapted version, does not rely on optimization, but outperformed other 'buffer insertion' heuristics in the study cited. The algorithm proceeds as follows: for a full order A on N, that is input to the algorithm, the starting time of an activity i is the integer nearest to $\delta_i(A)\omega$, with

$$\delta_i(A) = \frac{\sum_{\substack{(j,k)\in A:\\ (k,i)\in A \vee k=i}} p_j E_{L_j} L_j c_k}{\sum_{\substack{(j,k)\in A:\\ (k,i)\in A \vee k=i}} p_j E_{L_j} L_j c_k + \sum_{\substack{(j,k)\in A:\\ (i,j)\in A \vee i=j}} p_j E_{L_j} L_j c_k}.$$

If $(i, j) \in A$ then $\delta_i(A) \leq \delta_j(A)$, so that $s_i \leq s_j$, and we also have $\omega \geq s_i$ for every $i \in N$, since $\delta_i \in [0; 1]$, so the resulting schedule is feasible.

The foregoing results in four heuristics *HEA*, *HEN*, *HIA* and *HIN*, in which the second and third letter identify the sequencing and the scheduling method applied, respectively.

5 Computational experiments

In this section we discuss the experimental setup of our computational experiments (Sect. 5.1), we provide some figures to illustrate the computational performance of our B&B-algorithm (Sect. 5.2), and we compare the optimal solutions to our model with the output of the heuristics (Sect. 5.3).

5.1 Experimental setup

To examine the performance of the B&B-algorithm presented in Sects. 3.1–3.4 and the underlying single-disruption model, a series of computational experiments using randomly generated test problems has been conducted. Our implementation takes all integer inputs. Since the values p_i and g_{ik} may be fractional, primal objective-function coefficients are multiplied by the factor 10 000 and rounded to the lower integer. The coding was performed in C using the Microsoft Visual C++ 6.0 programming environment, and the experiments were run on a Dell Latitude D800 portable computer with a Pentium M processor with 1400 MHz clock speed and 512 MB RAM, equipped with Windows XP.

The experimental design adopted for this study consists of datasets of 25 problem instances involving $n = 8, 12, 16$ and 20 jobs. For each job i, we directly generate p_i-values rather than π_i. Half of the jobs in each instance have uncertain duration. For each such uncertain job i, a value q_i is selected from the discrete domain [1; 10] and these values are normalized to probabilities p_i. Cost coefficients c_i are integer values randomly selected from [0; 5]. For each activity i in each instance of these datasets the disturbance length L_i is a discrete random variable for which g_i is a discretization of the linear function $h_i(x) = 2(1/I_i - x/I_i^2)$, for which the intercept I_i with the abscissa is a realization of a discrete uniform random variable with support [2; 25]. Scenarios $k \in \Psi_i$ are determined as follows: l_{i1} is randomly selected from the discrete values in $[1; \min\{4, I_i - 1\}]$ and additional scenarios $l_{ik} = l_{i,k-1} + 5$ are added while $l_{ik} \leq I_i - 1$.

5.2 Computational performance

For the dataset with $n = 16$ the behavior of the B&B-algorithm is examined in Fig. 3 for ω varying from 0 to 42. We observe that the computational effort in terms of seconds of CPU-time is largest for ω ranging from 12 to 18 and then

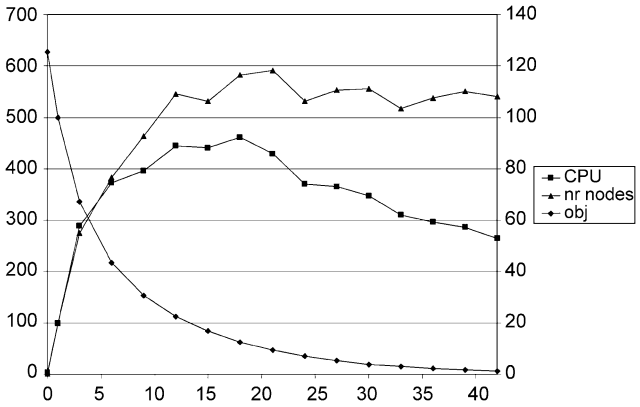

Fig. 3 The evolution of the number of nodes and the CPU-time (*left ordinate*) and the optimal objective function (*right ordinate*) as a function of ω, expressed in percentage points compared with case $\omega = 1$ ($n = 16$)

decreases with increasing ω. At the same time, the number of nodes in the search tree more or less stabilizes from $\omega = 12$ onwards. This phenomenon may be caused by the fact that the MCNFP-computations consume less time because the objective function to be reached is simply lower, while the number of examined solutions remains approximately the same. The same graph also provides an indication of the evolution of the optimal objective-function value. A significant stability gain can be achieved with moderate float values: the objective function falls steeply in the left part of the graph, and then levels off. Similar observations apply for other n-values.

The IP-formulation (7–10) (with dominance rule 2 enforced) was passed to the Lindo-solver (Industrial Lindo/PC release 6.01 (1997); the associated dynamic link library (dll) is called from our C-code). The running times of the solver ("IP") and of our algorithm ("B&B") are provided in Table 2 for different values for ω. The trends are obvious: we are able to produce optimal solutions to SWOD in considerably less computation time. It is likely that the IP-approach can be improved since we have only considered a straightforward implementation of the basic formulation, but we conclude that our B&B-code delivers at least competitive performance.

For the B&B-algorithm *without* dominance rule 1, Table 3 gives the CPU-times and the number of nodes expressed as a percentage of the corresponding values in Table 2. Clearly, this dominance rule is especially valuable for small idle times, and its usefulness increases with n.

5.3 Objective-function comparison with heuristics

Figure 4 summarizes the results of our comparisons with the benchmark heuristics. All simulations are based on independent job-duration distributions with parameters π_i, which are derived from the generated p_i-values in order to correspond to a specified expected number of disruptions $R = E[K] = \sum_{i \in N} \pi_i$. Higher R-values correspond to more variability in the system. From Sect. 2.5 we derive that

$$\pi_i = \frac{p_i}{\frac{P[0]}{P[1]} + p_i}, \quad i = 1, \ldots, n.$$

Every π_i is monotone decreasing in $\frac{P[0]}{P[1]}$. A good initial guess for a given value of R is $(\frac{P[0]}{P[1]})^{(0)} = (\frac{1}{R} - \frac{1}{n})$, which is exact if all p_i are equal ($p_i = 1/n$). This results in $R^{(0)} = \sum_{i \in N} \pi_i^{(0)}$, with $\pi_i^{(0)}$ being the corresponding first guess for the π_i. The numerical method regula falsi is applied to produce better estimates until the desired R is reached within a margin of at most 0.01 (three iterations are usually sufficient).

Compared with the B&B-algorithm, the running times are negligible for all four heuristics. Per scheduling instance and corresponding schedule, we estimated the expected weighted deviation by averaging the objective of 50 000 simulation runs. All results in this section pertain to the test set with $n = 16$ (results for other values of n exhibit comparable behavior). A vertical line in Fig. 4 indicates that for a particular problem instance the reference schedule SWOD has an objective-function value of 0 for higher ω-values, and so this instance is not included in the results to the right.

Case $R = 1$ (low variability, one activity disrupted *on average*) is closest to the one-disruption assumption of SWOD, and the model does indeed outperform the heuristics in this setting. Additionally, the introduction of a higher degree of uncertainty has only a limited effect on the results: SWOD still does considerably better than the heuristics, especially for large float values. We conclude that the output of SWOD is quite robust to deviations from the one-disruption assumption.

For low ω-values, the sequencing approach is the key performance determinant: *HEN* and *HEA* cross the ordinate at 100% (the *EWDL*-rule is optimal for $\omega = 0$), versus almost 200% for *HIN* and *HIA*. From comparison of *HEN* with *HEA* and *HIN* with *HIA*, it can be concluded that the use of the network-flow technique for buffer insertion is valuable for an arbitrary input job sequence.

6 Summary and conclusions

In the field of scheduling under uncertainty, the *stability* objective is a new topic. This paper has examined the development of a stable one-machine schedule, in which small changes due to job-duration fluctuations have only a local effect and do not propagate throughout the scheduling horizon. Deterministic schedules were proposed with explicitly inserted idle time serving as protective buffer time. We have

Table 2 Average CPU-times for the B&B-algorithm and the Lindo solver, for $n = 8, 12, 16, 20$. The settings without entry were not run to termination because of excessive computation time

n	$\omega = 1$		$\omega = 3$		$\omega = 9$		$\omega = 15$	
	B&B	IP	B&B	IP	B&B	IP	B&B	IP
8	0.00 s	0.14 s	0.00 s	0.14 s	0.00 s	0.12 s	0.00 s	0.11 s
12	0.01 s	25.9 s	0.02 s	27.6 s	0.02 s	29.1 s	0.02 s	29.4 s
16	0.10 s	>1000 s	0.37 s	>1000 s	0.47 s	>1000 s	0.62 s	>1000 s
20	3.37 s	–	32.29 s	–	111.86 s	–	146.49 s	–

Table 3 Average CPU-times and number of examined search nodes for the B&B-algorithm without dominance rule 1, expressed as a percentage of the values obtained by the final version of the algorithm

n	$\omega = 1$		$\omega = 3$		$\omega = 9$		$\omega = 15$	
	CPU	Nodes	CPU	Nodes	CPU	Nodes	CPU	Nodes
12	117.5%	127.3%	109.3%	105.2%	98.7%	100.5%	91.8%	100.0%
16	204.3%	195.1%	106.1%	114.1%	101.3%	101.4%	103.1%	100.2%
20	815.0%	678.4%	160.9%	164.5%	102.7%	105.4%	99.8%	101.0%

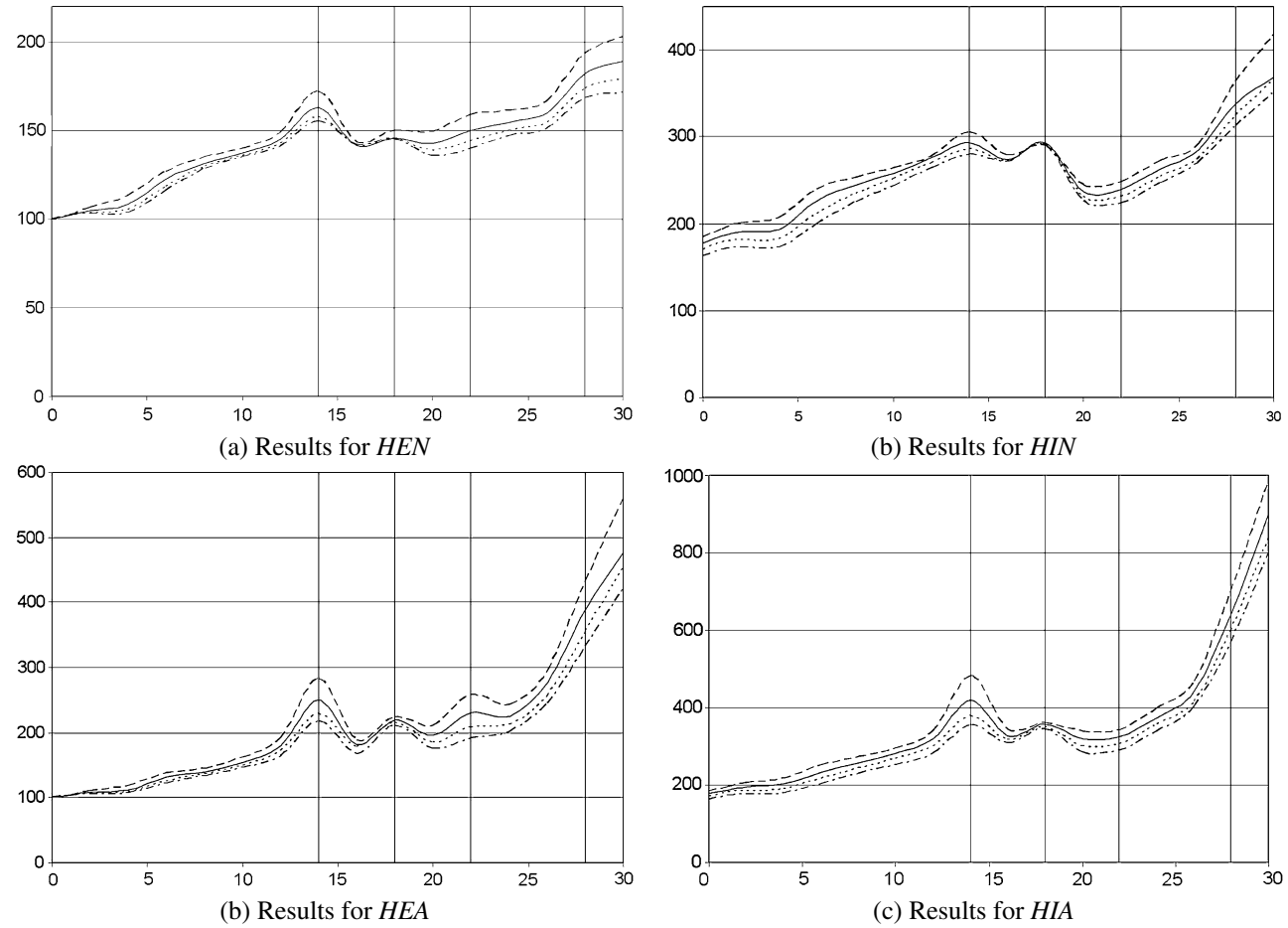

Fig. 4 For each heuristic, the corresponding graph represents the objective function resulting from simulation with independent job durations, expressed in percentage points compared to the optimal SWOD-schedule. ω is on the abscissa. The *four curves* (*highest to lowest*) correspond with $R = 1, 2, 3$ and 4, respectively. Each *vertical line* indicates exclusion of one problem instance for all observations to the *right of the line*.

observed that a significant stability gain can be achieved with moderate buffer sizes; the incremental benefit of increasing the buffers has been decreasing.

A mathematical-programming model was presented to minimize the expected weighted deviation in starting times of the jobs when *exactly one* job suffers a deviation from its baseline duration. This model only considers the main effects of the separate disruption of each of the jobs rather than all possible disruption interactions. The model was shown to yield consistently good results for a wide range of variability settings.

The branch-and-bound procedure that we have developed to solve the proposed model is several orders of magnitude faster than a general IP-solver. The size of the scheduling instances that can be solved to guaranteed optimality remains limited but is comparable with the size of problems solvable by other combinatorial-optimization approaches to scheduling under uncertainty (see, e.g., Daniels and Carrillo 1997; Daniels and Kouvelis 1995; or Kouvelis et al. 2000). An additional complication of the stability objective is the fact that optimal schedules need not (and generally will not) be active. Further research is needed if stable schedules are to be developed for realistically sized scheduling problems; we are convinced that the insights provided in this paper can serve as guidelines in this process. It is worth noting that the incorporation of precedence constraints between jobs reduces the size of the search space and may actually render algorithms more efficient; the same may hold for other additional constraints on feasible schedules.

Each job's baseline duration is currently its minimum possible duration; this choice by itself constitutes a topic for further research (related to the issue of duration estimation, which is studied by Britney 1976, amongst others). Finally, one particular variation of the 'no early start' assumption can be suggested that would probably enhance the practical validity of the proposed model, namely the generalization towards time buckets that make up the scheduling horizon, and in which each job is available for execution at the earliest at the start of its assigned time bucket.

Acknowledgements We thank the two anonymous referees for their constructive comments.

This research has been partially supported by project OT/03/14 of the Research Fund K.U.Leuven and project G.0109.04 of the Research Programme of the Research Foundation, Flanders (Belgium) (F.W.O.-Vlaanderen). Roel Leus is partly funded as Postdoctoral Fellow of the Research Foundation, Flanders.

Appendix: Counterexample for convexity of lb_1

We examine the behavior of lb_1 for a scheduling problem with $N \backslash J_h = \{1, 2\}$ containing two jobs, as a function of f for $f = 0$ to $f = 3$, if $c_1 = c_2 = 1$, $p_1 = 0.99$, $p_2 = 0.01$, $\Psi_1 = \{1, 2, 3\}$, $\Psi_2 = \{50, 51, 52\}$, and all g_{ik} equal. For successive values of $f = 0, 1, 2, 3$, we have $lb_1 = 0.51, 0.5, 0.33, 0$, such that the speed of descent increases with f, and the function cannot be convex.

References

Adiri, I., Bruno, J., Frostig, E., & Rinnooy Kan, A. H. G. (1989). Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, *36*, 679–696.

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows*. New York: Prentice-Hall.

Akturk, M., & Gorgulu, E. (1999). Match-up scheduling under a machine breakdown. *European Journal of Operational Research*, *112*, 81–97.

Aytug, H., Lawley, M., McKay, K., Mohan, S., & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: a review and some future directions. *European Journal of Operational Research*, *161*, 86–110.

Bean, J., Birge, J., Mittenthal, J., & Noon, C. (1991). Match-up scheduling with multiple resources, release dates and disruptions. *Operations Research*, *39*, 470–483.

Britney, R. R. (1976). Bayesian point estimation and the PERT scheduling of stochastic activities. *Management Science*, *22*, 938–948.

Bruno, J., Coffmann, E., & Sethi, R. (1974). Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, *17*, 382–387.

Calhoun, K., Deckro, R., Moore, J., Chrissis, J., & Hove, J. V. (2002). Planning and re-planning in project and production planning. *Omega*, *30*, 155–170.

Christy, D. P., & Kanet, J. J. (1990). Manufacturing systems with forbidden early shipment: implications for choice of scheduling rules. *International Journal of Production Research*, *28*, 91–100.

Daniels, R., & Carrillo, J. (1997). β-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, *29*, 977–985.

Daniels, R., & Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, *41*, 363–376.

Dasdan, A., & Gupta, R. (1998). Faster maximum and minimum mean cycle algorithms for system performance analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *17*, 889–899.

Elmaghraby, S. E. (2005). On the fallacy of averages in project risk management. *European Journal of Operational Research*, *165*, 307–313.

Fearnhead, P., & Meligkotsidou, L. (2004). Exact filtering for partially observed continuous time models. *Journal of the Royal Statistical Society: Series B*, *66*, 771–789.

Herroelen, W., & Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research*, *156*, 550–565.

Kanet, J. J., & Christy, D. P. (1984). Manufacturing systems with forbidden early departure. *International Journal of Production Research*, *22*, 41–50.

Kanet, J., & Sridharan, V. (2000). Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research*, *48*, 99–110.

Karp, R. (1978). A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, *23*, 309–311.

Kouvelis, P., & Yu, G. (1997). *Robust discrete optimization and its applications*. Dordrecht: Kluwer Academic.

Kouvelis, P., Daniels, R. L., & Vairaktarakis, G. (2000). Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, *32*, 421–432.

Leon, V., Wu, S., & Storer, R. (1994). Robustness measures and robust scheduling for job shops. *IIE Transactions*, *26*, 343–362.

Leus, R. (2003). *The generation of stable project plans. Complexity and exact algorithms*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium.

Leus, R., & Herroelen, W. (2005). The complexity of machine scheduling for stability with a single disrupted job. *Operations Research Letters*, *33*, 151–156.

Mehta, S., & Uzsoy, R. (1998). Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, *14*, 365–378.

O'Donovan, R., Uzsoy, R., & McKay, K. (1999). Predictable scheduling on a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, *37*, 4217–4233.

Parker, R., & Rardin, R. (1988). *Discrete optimization*. New York: Academic.

Pinedo, M. (2002). *Scheduling. Theory, algorithms, and systems*. New York: Prentice-Hall.

Raheja, A., & Subramaniam, V. (2002). Reactive recovery of job shop schedules—a review. *International Journal of Advanced Manufacturing Technology*, *19*, 756–763.

Rangsaritratsamee, R., Ferrel, W., & Kurz, M. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers & Industrial Engineering*, *46*, 1–15.

Stork, F. (2001). *Stochastic resource-constrained project scheduling*. PhD thesis, TU Berlin, Berlin, Germany.

Wu, S., Storer, H., & Chang, P.-C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research*, *20*, 1–14.

Yáñez, J., & Ramírez, J. (2003). The robust coloring problem. *European Journal of Operational Research*, *148*, 546–558.

Yano, C. A. (1987). Setting planned leadtimes in serial production systems with tardiness costs. *Management Science*, *33*, 95–106.