3

Relaxation-Based Algorithms

Relaxation-based algorithms for resource-constrained project scheduling with regular or convexifiable objective functions rely on the first basic representation of the set $\mathcal{S}$ of all feasible schedules as a union of relation polytopes. By deleting the resource constraints we obtain the resource relaxation, which coincides with the time-constrained project scheduling problem. The latter problem can be solved efficiently by computing the minimal point ES of set $\mathcal{S}_T$ if f is regular or some local minimizer of the objective function f in set $\mathcal{S}_T$ if f is convexifiable. Clearly, the tractability of the problem is preserved when moving from set $\mathcal{S}_T$ to arbitrary nonempty relation polytopes $\mathcal{S}_T(\rho)$. Starting with the resource relaxation, i.e., with the empty relation, relaxation-based algorithms iteratively put the resource constraints into force by branching over time-feasible extensions ρ' of the respective parent relation ρ. Each relation ρ' defines a collection of precedence constraints that break up some forbidden active set $\mathcal{A}(S,t)$ belonging to a minimizer S of f on search space $\mathcal{P} = \mathcal{S}_T(\rho)$. The branching process is continued until either $\mathcal{S}_T(\rho) = \emptyset$ or the minimizer S of f on $\mathcal{S}_T(\rho)$ is feasible. The latter condition is necessarily satisfied as soon as relation ρ is feasible. Note, however, that schedule S may be feasible even before ρ has been extended to a feasible relation. When dealing with regular objective functions, the ordinary precedence constraints given by relations ρ may be replaced by disjunctive precedence constraints (cf. Subsections 1.2.3 and 1.3.3). Since a disjunctive precedence constraint corresponds to the disjunction of several ordinary precedence constraints, branching is then performed over sets of relations and consequently, the search spaces $\mathcal{P}$ on which f is to be minimized represent unions of relation polytopes.

From now on we assume that the project under consideration comprises renewable and cumulative resources, where the renewable resources are used by real activities $i \in V^a$ and the cumulative resources are depleted and replenished by events $i \in V^e$. Accordingly, for given schedule S the active sets

$$\mathcal{A}(S,t) := \{i \in V^a \mid S_i \leq t < S_i + p_i\} \cup \{i \in V^e \mid S_i \leq t\}$$

at times t contain both real activities and events, and resource-feasible schedules satisfy both the renewable-resource constraints (1.7) and the cumulative-resource constraints (1.20). The set of all feasible schedules is now $\mathcal{S} = \mathcal{S}_T \cap \mathcal{S}_R \cap \mathcal{S}_C$. As a straightforward extension of the definitions from Subsections 2.1.1 and 2.1.2, we say that a relation ρ *in set* V is time-feasible if $\mathcal{S}_T(\rho) \neq \emptyset$ and is feasible if $\emptyset \neq \mathcal{S}_T(\rho) \subseteq \mathcal{S}$. It is easily seen that first, relation ρ is again time-feasible precisely if relation network $N(\rho)$ does not contain any cycle of positive length and that second, a time-feasible relation ρ is feasible exactly if both induced sub-relations $\rho \cap (V^a \times V^a)$ and $\rho \cap (V^e \times V^e)$ are feasible in the sense of Definitions 2.3 and 2.17. As a consequence of the latter statement, the feasibility of a time-feasible relation ρ in set V can be verified by sequentially applying the network flow techniques discussed in Subsections 2.1.1 and 2.1.2 to the respective sub-relations.

The *resource-constrained project scheduling problem* to be dealt with reads as follows:

$$\left.\begin{array}{ll} \text{Minimize} & f(S) \\ \text{subject to} & S \in \mathcal{S}_T \cap \mathcal{S}_R \cap \mathcal{S}_C \end{array}\right\} \quad \text{(P)}$$

where f is some regular or convexifiable objective function. In Section 3.1 we treat the case of regular objective functions. Section 3.2 is devoted to convexifiable objective functions.

3.1 Regular Objective Functions

We first develop an enumeration scheme based on the concept of disjunctive precedence constraints that either generates a set of candidate schedules containing an optimal schedule or proves that there is no feasible schedule for the project under consideration. We are then concerned with the relaxation to be solved at each enumeration node. The latter problem amounts to minimizing a regular objective function subject to temporal and disjunctive precedence constraints. Next, we discuss the extension of the enumeration scheme to a branch-and-bound algorithm and review alternative solution procedures for resource-constrained project scheduling with regular objective functions.

3.1.1 Enumeration Scheme

In this subsection we are concerned with an enumeration scheme for problem (P) with regular objective function f which forms the basis of branch-and-bound procedures by Schwindt (1998*a*) and Neumann and Schwindt (2002) for solving the project duration problem with renewable or cumulative resources, respectively. Consider an optimal solution S to the time-constrained project scheduling problem (1.2) with a regular objective function f, e.g., $S = ES = \min \mathcal{S}_T$. If S satisfies the renewable-resource constraints (1.7) and

the cumulative-resource constraints (1.20), S is an optimal schedule. Otherwise, there is some point in time $t \in [0, \overline{d}]$ such that $F := \mathcal{A}(S,t) \cap V^a$ or $F := \mathcal{A}(S,t) \cap V^e$ represents a forbidden set. In the former case, the joint requirements by real activities $i \in F$ exceed the capacity of some renewable resource $k \in \mathcal{R}^\rho$, and in the latter case, the depletions and replenishments by events $i \in F$ create a surplus or a shortage in some cumulative resource $k \in \mathcal{R}^\gamma$. Forbidden set F can be broken up by introducing a disjunctive precedence constraint (see Subsections 1.2.3 and 1.3.3)

$$\min_{j \in B} S_j \geq \min_{i \in A}(S_i + p_i) \tag{3.1}$$

between some appropriate set A and a minimal delaying alternative B, where by definition $p_i = 0$ for $i \in V^e$. If resource k is renewable, we choose $A := F \backslash B$. Otherwise, we put $A := V_k^{e^-} \setminus F$ if F is a k-surplus set and $A := V_k^{e^+} \setminus F$ if F is a k-shortage set. Let

$$P(A,B) := \bigcup_{i \in A} \{\{i\} \times B\}$$

denote the set of irreflexive relations $\{i\} \times B$ with $i \in A$, which each give rise to the (ordinary) precedence constraints between activity i and all activities $j \in B$. Introducing disjunctive precedence constraint (3.1) refines the resource relaxation by restricting the initial search space $\mathcal{P} = \mathcal{S}_T$ to the set of all schedules S contained in the union of relation polytopes $\mathcal{S}_T(\rho)$ with $\rho \in P(A,B)$.

After the selection of a minimal delaying alternative B, we minimize f on the restricted search space. Checking the resource-feasibility of the resulting minimizer, refining the relaxation by disjunctive precedence constraints, and re-optimizing f on the restricted search space is performed until either the search space has become void or the resulting minimizer S of f is resource-feasible. The disjunctive precedence constraints are represented as a collection P of relations ρ whose relation polytopes $\mathcal{S}_T(\rho)$ cover the search space. In each iteration, when adding a disjunctive precedence constraint of type (3.1) we put $P := P \otimes P(A,B)$ where $P = \{\emptyset\}$ at the root node and

$$P \otimes P(A,B) := \bigcup_{\rho \in P, \rho' \in P(A,B)} \{\rho \cup \rho'\}$$

As we shall see in Subsection 3.1.2, each of the nonempty search spaces $\mathcal{P} = \cup_{\rho \in P} \mathcal{S}_T(\rho)$ generated in this way possesses a unique minimal point, which represents a minimizer S of f on set $\mathcal{P}$.

We now consider the enumeration scheme in more detail. The corresponding procedure is given by Algorithm 3.1. Let Q denote a list of relation sets P in set V and let $\mathcal{C}$ designate the set of candidate schedules generated. Starting with $Q = \{\{\emptyset\}\}$ and $\mathcal{C} = \emptyset$, at each iteration we remove some relation set P from Q and solve the relaxation by either computing the minimal point S of search space $\mathcal{P} = \cup_{\rho \in P} \mathcal{S}_T(\rho)$ or showing that $\mathcal{P} = \emptyset$. In the latter case,

we write $S = S^{\infty} := (\infty, \ldots, \infty)$. For $S < S^{\infty}$, we proceed as follows. If schedule S is resource-feasible, we have found a candidate schedule and put $\mathcal{C} := \mathcal{C} \cup \{S\}$. Otherwise, there is a start time $t = S_i$ of some activity $i \in V$ such that active set $\mathcal{A}(S,t)$ includes a forbidden set of real activities or a forbidden set of events. In the former case, we compute the set $\mathcal{B}$ all minimal delaying alternatives B for $F := \mathcal{A}(S,t) \cap V^a$ by using Algorithm 1.4. Otherwise, $F := \mathcal{A}(S,t) \cap V^e$ is a k-surplus or a k-shortage set for some cumulative resource $k \in \mathcal{R}^\gamma$, and calling Algorithm 1.6 provides the set $\mathcal{B}$ of all minimal delaying alternatives for F and k. For each minimal delaying alternative $B \in \mathcal{B}$ we then introduce disjunctive precedence constraint (3.1) between the corresponding set A and set B by setting $P' := P \otimes P(A,B)$ and adding the expanded relation set P' on list Q. We return to the (refined) relaxation and reiterate these steps until all relation sets P in list Q have been investigated, i.e., until $Q = \emptyset$. Finally, we return the set $\mathcal{C}$ of all candidate schedules found.

Algorithm 3.1. Enumeration scheme for regular objective functions

Input: A project.
Output: Set $\mathcal{C}$ of candidate schedules.

initialize list of relation sets $Q := \{\{\emptyset\}\}$ and set of candidate schedules $\mathcal{C} := \emptyset$;
repeat
  delete some relation set P from list Q;
  determine schedule $S = \min(\cup_{\rho \in P} \mathcal{S}_T(\rho))$;
  if $S < S^{\infty}$ **then** (∗ search space is nonempty ∗)
    if S is resource-feasible **then** $\mathcal{C} := \mathcal{C} \cup \{S\}$; (∗ candidate schedule found ∗)
    else (∗ introduce disjunctive precedence constraints ∗)
      determine time t such that resource constraints (1.7) or (1.20) are violated for some $k \in \mathcal{R}^\rho \cup \mathcal{R}^\gamma$;
      if $k \in \mathcal{R}^\rho$ **then**
        set $F := \mathcal{A}(S,t) \cap V^a$;
        compute set $\mathcal{B}$ of all minimal delaying alternatives for F;
        (∗ Algorithm 1.4 ∗)
      else
        set $F := \mathcal{A}(S,t) \cap V^e$;
        compute set $\mathcal{B}$ of all minimal delaying alternatives for F and k;
        (∗ Algorithm 1.6 ∗)
      for all $B \in \mathcal{B}$ **do**
        if $k \in \mathcal{R}^\rho$ **then** set $A := F \setminus B$; **elsif** $B \subseteq V_k^{e+}$ **then** set $A := V_k^{e-} \setminus F$;
        else set $A := V_k^{e+} \setminus F$;
        set $P' := P \otimes P(A,B)$ and add P' on list Q;
until $Q = \emptyset$;
return $\mathcal{C}$;

The following proposition establishes the correctness of the enumeration scheme from Algorithm 3.1.

Proposition 3.1 (Neumann et al. 2003*b*, Sect. 2.5). *Let $\mathcal{C}$ be the set of candidate schedules generated by Algorithm 3.1 and let $\mathcal{OS}$ denote the set of all optimal schedules.*

(a) *Algorithm 3.1 is finite.*
(b) *Algorithm 3.1 is complete, i.e., $\mathcal{C} \cap \mathcal{OS} = \emptyset$ if and only if $\mathcal{S} = \emptyset$.*
(c) *All schedules generated by Algorithm 3.1 are quasiactive, i.e., $\mathcal{C} \subseteq \mathcal{QAS}$.*

Proof.

(a) At each iteration a relation set P is removed from list Q and a finite number of expanded relation sets P' are added to Q. For each $\rho \in P$, sets P' contain a relation $\rho' \supset \rho$ each. Since the cardinality of any irreflexive relation in set V is bounded from above by $(n+1)(n+2)$, this implies that the number of iterations performed by Algorithm 3.1 is finite.
(b) Clearly, the search space $\mathcal{P} = \mathcal{S}_T(\emptyset) = \mathcal{S}_T$ associated with the initial relation set $P = \{\emptyset\}$ is a superset of the feasible region $\mathcal{S}$ and thus $\mathcal{S} = \mathcal{S}_T(\emptyset) \cap \mathcal{S}$. Now let S be the minimal point of some search space $\mathcal{P}$ enumerated in the course of Algorithm 3.1. If S is not resource-feasible, there is a time t such that active set $\mathcal{A}(S,t)$ includes a forbidden set F. Let $\mathcal{B}$ be the set of minimal delaying alternatives for F. Then Theorems 1.17 and 1.28 say that any resource-feasible schedule in set $\mathcal{P}$ satisfies one of the disjunctive precedence constraints (3.1) with $B \in \mathcal{B}$ and appropriate set A. Since in addition all enumerated schedules S minimize f on the respective search spaces, there is at least one optimal candidate schedule $S \in \mathcal{C}$ provided that $\mathcal{S} \neq \emptyset$. Conversely, all candidate schedules $S \in \mathcal{C}$ are feasible and the lower semicontinuity of f implies that $\mathcal{OS} = \emptyset$ only if $\mathcal{S} = \emptyset$. Consequently, from $\mathcal{C} \neq \emptyset$ it follows that $\mathcal{OS} \neq \emptyset$.
(c) Each candidate schedule $S \in \mathcal{C}$ is the minimal point of some relation polytope $\mathcal{S}_T(\rho)$ and feasible. Due to $\theta(S) \supseteq \rho$ and thus $\mathcal{S}_T(\theta(S)) \subseteq \mathcal{S}_T(\rho)$, it follows from $S \in \mathcal{S}_T(\theta(S))$ that S is the minimal point of its schedule polytope $\mathcal{S}_T(\theta(S))$ as well, i.e., $S \in \mathcal{QAS}$. □

We notice that as a direct consequence of the proof of Proposition 3.1a, the maximum depth of the enumeration tree generated by Algorithm 3.1 is $\mathcal{O}(n^2)$. Moreover, the candidate schedules $S \in \mathcal{C}$ are generally not active but only quasiactive. Recall that already deciding on whether or not a given feasible schedule is active constitutes an NP-hard problem.

3.1.2 Solving the Relaxations

In this subsection we are concerned with the problem of minimizing a regular objective function f on a search space $\mathcal{P}$ defined by temporal constraints and disjunctive precedence constraints. We assume that the disjunctive precedence constraints are given by a collection of ν relation sets $P(A_1,B_1),\ldots,P(A_\nu,B_\nu)$ with $P(A_\mu,B_\mu) = \cup_{i\in A_\mu}(\{i\}\times B_\mu)$ for $\mu = 1,\ldots,\nu$.

With $P = \otimes_{\mu=1}^{\nu} P(A_\mu, B_\mu)$ we then have

$$\mathcal{P} = \cup_{\rho \in P} \mathcal{S}_T(\rho) = \cap_{\mu=1}^{\nu} \cup_{i \in A_\mu} \mathcal{S}_T(\{i\} \times B_\mu)$$

Proposition 3.2 (Neumann and Schwindt 2002). *Let ψ be the operator on partially ordered set $(\mathbb{R}_{\geq 0}^{n+2}, \leq)$ with $\psi(S) = (\psi_j(S))_{j \in V}$ and*

$$\psi_j(S) = \max\big(0, \max_{(i,j) \in E}(S_i + \delta_{ij}), \max_{\substack{\mu=1,\ldots,\nu:\\ j \in B_\mu}} \min_{i \in A_\mu}(S_i + p_i)\big) \quad (j \in V).$$

(a) *If $\mathcal{P} \neq \emptyset$, set $\mathcal{P}$ has a unique minimal point S^+.*
(b) *ψ possesses a fixed point if and only if $\mathcal{P} \neq \emptyset$. Minimal point S^+ coincides with the unique fixed point S of ψ with $S_0 = 0$.*
(c) *If $\mathcal{P} \neq \emptyset$, S^+ arises as the limit of the sequence $\{S^\lambda\}$ with $S^1 = ES$ and $S^{\lambda+1} = \psi(S^\lambda)$ for $\lambda \in \mathbb{N}$.*
(d) *If $\mathcal{P} \neq \emptyset$, there is a $\kappa \leq n\overline{d}$ with $\psi(S^\lambda) = S^\lambda = S^+$ for all $\lambda \geq \kappa$.*

Proof.

(a) Let S^+ be the schedule given by $S_j^+ := \min_{S \in \mathcal{P}} S_j$ for all $j \in V$ and assume that $\mathcal{P} \neq \emptyset$. We show that S^+ is the unique minimal point of $\mathcal{P}$. Since ψ is isotonic and $S^+ \leq S$ holds for all $S \in \mathcal{P}$, we have $\psi(S^+) \leq \min_{S \in \mathcal{P}} \psi(S)$. By definition of ψ, set $\mathcal{P}$ can be represented as $\mathcal{P} = \{S \in \mathbb{R}_{\geq 0}^{n+2} \mid S_0 = 0,\ S \geq \psi(S)\}$. Now assume that $S^+ \notin \mathcal{P}$. Then there is an activity $j \in V$ such that $S_j^+ < \psi_j(S^+) \leq \min_{S \in \mathcal{P}} \psi_j(S) \leq \min_{S \in \mathcal{P}} S_j = S_j^+$, which contradicts the assumption.

(b) Since S^+ is *componentwise* minimal in set $\mathcal{P} = \{S \in \mathbb{R}_{\geq 0}^{n+2} \mid S_0 = 0, S \geq \psi(S)\}$, we have $S^+ = \psi(S^+)$, i.e., S^+ is a fixed point of ψ. Due to the connectivity of network N, a point S is a fixed point of ψ exactly if there is an $\alpha \geq 0$ with $S = S^+ + \alpha(1, \ldots, 1)$. Thus, $S = S^+$ is the unique fixed point of ψ with $S_0 = 0$. Now assume that $\mathcal{P} = \emptyset$. Then there is no point $S \in \mathbb{R}_{\geq 0}^{n+2}$ such that $S_0 = 0$ and $S \geq \psi(S)$. Since the set of fixed points of ψ equals $\{S \in \mathbb{R}_{\geq 0}^{n+2} \mid S = S^+ + \alpha(1, \ldots, 1)$ for some $\alpha \geq 0\}$ and $S_0^+ = 0$, the latter statement implies that ψ does not possess any fixed point.

(c) We first show by induction on λ that $S^\lambda \leq S^+$ for all $\lambda \in \mathbb{N}$. From $\mathcal{P} \subseteq \mathcal{S}_T$ it follows that $S^1 = ES = (\min_{S \in \mathcal{S}_T} S_j)_{j \in V} \leq (\min_{S \in \mathcal{P}} S_j)_{j \in V} = S^+$. Now assume that $S^\lambda \leq S^+$. Since operator ψ is isotonic, we have $S^{\lambda+1} = \psi(S^\lambda) \leq \psi(S^+) \leq S^+$, where the last inequality results from $S^+ \in \mathcal{P}$. For $S^1 = ES$ it holds that $S_j^1 = [\max_{(i,j) \in E}(S_i^1 + \delta_{ij})]^+$ for all $j \in V$. This provides $S^2 = \psi(S^1) \geq S^1$, which proves the sequence $\{S^\lambda\}$ to be componentwise nondecreasing. Thus, the existence of an upper bound S^+ implies the convergence of $\{S^\lambda\}$. Then $\lim_{\lambda \to \infty} S^\lambda = \lim_{\lambda \to \infty} S^{\lambda+1} = \lim_{\lambda \to \infty} \psi(S^\lambda) = \psi(\lim_{\lambda \to \infty} S^\lambda)$, and the limit of $\{S^\lambda\}$ represents a fixed point of ψ. The last equation is due to the continuity of ψ. From $S_0^1 = ES_0 = 0 \leq \lim_{\lambda \to \infty} S_0^\lambda \leq S_0^+ = 0$ we obtain $\lim_{\lambda \to \infty} S_0^\lambda = 0$. Since $S = S^+$ is the unique fixed point S of ψ with $S_0 = 0$, S^+ coincides with the limit of sequence $\{S^\lambda\}$.

(d) The assertion is immediate with the monotonicity of $\{S^\lambda\}$ and the property that as long as $S^{\lambda+1} \neq S^\lambda$, there is an activity $h \in V$ with $S_h^{\lambda+1} \geq S_h^\lambda + 1$. □

According to Proposition 3.2, minimizing a regular objective function f on set $\mathcal{P}$ can be achieved by starting with $S = ES$ and putting $S := \psi(S)$ until either $S = \psi(S)$ or $S_{n+1} > \overline{d}$. In the latter case, $\mathcal{P}$ has been shown to be empty. The number of iterates needed for reaching minimal point S^+ can be decreased by the following modifications. First, we may start the procedure with any time-feasible schedule $S \leq S^+$. Second, each time the start time S_j of some activity j has been increased due to a disjunctive precedence constraint, we may immediately restore the time-feasibility of schedule S by putting $S_h := \max(S_h, S_j + d_{jh})$ for all $h \in V$. The resulting schedule is time-feasible precisely if $S_{n+1} \leq \overline{d}$, which is easily seen by adding arc $(0, j)$ with weight $\delta_{0j} = S_j$ to project network N and applying Algorithm 1.3 for updating distance matrix D (see Remark 1.8). For $\mu = 1, \ldots, \nu$ let $t_\mu := \min_{i \in A_\mu}(S_i + p_i)$ be the earliest completion time of some activity $i \in A_\mu$ with respect to current iterate $S \in \mathcal{S}_T$ and let $d_\mu^h := \max_{j \in B_\mu} d_{jh}$ denote the "distance" between set B_μ and activity $h \in V$. Then the start time of activity $h \in V$ has to be increased precisely if $S_h < t_\mu + d_\mu^h$. In this case, we set $S_h := t_\mu + d_\mu^h$ and update the earliest completion times t_λ for all sets A_λ containing activity h. Algorithm 3.2 shows an implementation of this method as a label-correcting procedure where queue Q contains all indices $\lambda = 1, \ldots, \nu$ for which time t_λ has to be updated.

Algorithm 3.2. Minimizing regular objective functions subject to temporal and disjunctive precedence constraints

---

Input: A schedule $S' \in \mathcal{S}_T$, distance matrix D, relation sets $P(A_\mu, B_\mu)$ $(\mu = 1, \ldots, \nu)$.
Output: Minimal schedule $S \geq S'$ in set $\mathcal{P} = \bigcap_{\mu=1}^{\nu} \bigcup_{i \in A_\mu} \mathcal{S}_T(\{i\} \times B_\mu)$.

```
    set S := S' and Q := {1, ..., ν};
    for all μ = 1, ..., ν do
3:      t_μ := min_{i∈A_μ}(S_i + p_i);
4:      for all h ∈ V do d_μ^h := max_{j∈B_μ} d_jh;
    repeat
      dequeue index μ from Q;
7:    for all h ∈ V with S_h < t_μ + d_μ^h do
8:      set S_h := t_μ + d_μ^h;
        for all λ = 1, ..., ν with h ∈ A_λ and t_λ < min_{g∈A_λ}(S_g + p_g) do
10:       set t_λ := min_{g∈A_λ}(S_g + p_g);
          if λ ∉ Q then enqueue λ to Q;
    until Q = ∅ or S_{n+1} > d̄;
    if Q = ∅ then return S; else return S^∞;
```

Next we analyze the time complexity of Algorithm 3.2. To this end, we assume that sets V and A_μ with $\mu = 1, \ldots, \nu$ are stored as Fibonacci heaps (see, e.g., Knuth 1998, Sect. 6.2) sorted respectively according to nondecreasing start times S_i or nondecreasing completion times $S_i + p_i$. The initialization of earliest completion times t_μ and distances d_μ^h on lines 3 and 4 takes $\mathcal{O}(\nu n^2)$ time. Since the algorithm stops as soon as $S_{n+1} > \overline{d}$, line 8 is executed at most $\mathcal{O}(n\overline{d})$ times. On the other hand, on line 10 each point in time t_λ cannot be increased more than $\mathcal{O}(\nu\overline{d})$ times, which implies that the repeat-loop is iterated $\mathcal{O}(\min[n, \nu]\overline{d})$ times. At each iteration, identifying activities $h \in V$ with $S_h < t_\mu + d_\mu^h$ on line 7 requires $\mathcal{O}(\log n)$ time and rearranging the Fibonacci heaps V and A_μ after having increased start time S_h on line 8 takes $\mathcal{O}(\nu \log n)$ time. Thus, the time complexity of Algorithm 3.2 is $\mathcal{O}(\nu n^2 + \min[n, \nu]\overline{d}\nu \log n)$.

Alternative solution procedures with pseudo-polynomial time complexity have been devised by Zwick and Paterson (1996), Chauvet and Proth (1999), and Schwiegelshohn and Thiele (1999). Möhring et al. (2004) provide a review on papers dealing with applications of disjunctive precedence constraints that arise in fields outside project scheduling, such as analyzing functional dependencies among data in relational data bases (Ausiello et al. 1983), optimizing the partial disassembly of products when removing single components (Goldwasser and Motwani 1999), or computing optimal strategies for mean-payoff games on directed bipartite graphs (Zwick and Paterson 1996). In the latter paper it is shown that the problem to decide whether the outcome of such a game is positive is contained in $NP \cap coNP$. In addition, Möhring et al. (2004) have shown that this decision problem is polynomially equivalent to minimizing a regular objective function subject to disjunctive temporal constraints where p_i in inequality (1.11) is replaced with an arbitrary time lag δ_{ij}. Despite this observation, however, no algorithm is available thus far for solving the latter scheduling problem in polynomial time. For the case where all time lags δ_{ij} are nonnegative, Möhring et al. (2004) exhibit a label-setting algorithm that runs in $\mathcal{O}(n + [\sum_{\mu=1}^{\nu} |B_\mu|][m + \sum_{\mu=1}^{\nu} |A_\mu||B_\mu|])$ time.

3.1.3 Branch-and-Bound

The enumeration scheme given by Algorithm 3.1 defines the **branching strategy** of a branch-and-bound algorithm for problem (P) with regular objective function f. In this subsection we present the complete branch-and-bound procedure. Besides the branching strategy, a branch-and-bound algorithm for a minimization problem is characterized by the *search strategy* for selecting one of the generated enumeration nodes for further branching, *consistency tests*, which are applied to restrict the search spaces of enumeration nodes, and *lower bounds* on the minimum objective function value.

The **search strategy** of the branch-and-bound algorithm is as follows. We always branch from one of the child nodes v of the node u currently selected, i.e., we perform a *depth-first search*. The depth-first strategy can be implemented by simply choosing the list Q of unexplored nodes u to be a

stack. The main advantages of depth-first search are that first, this strategy minimizes the memory requirements necessary for storing list Q and that second, the number of branchings for reaching the first leaf u of the enumeration tree (and thus often the time for computing a first feasible solution) is minimum. Child nodes v are pushed onto stack Q according to nonincreasing lower bounds. One drawback of the depth-first search strategy is that typically, two enumeration nodes visited consecutively belong to similar relation sets, which share a large number of common elements. As a consequence, it may take a long time before any schedule located in a given part of the feasible region is investigated, and thus the algorithm may spend much time in useless parts of the enumeration tree. This shortcoming can be avoided by partitioning the enumeration tree into a number of subtrees, which are simultaneously traversed according to a depth-first search strategy each (*scattered search*, cf. Klein and Scholl 2000).

Basically, each of the **consistency tests** discussed in Subsections 1.2.4 and 1.3.4 can be applied at any enumeration node. Since disjunctive precedence constraints cannot be represented by a distance matrix D, the tests using distances d_{ij} between arbitrary nodes $i, j \in V$ (the disjunctive activities, energy precedence, and balance tests) refer to a modified distance matrix $D' = (d'_{ij})_{i,j \in V}$ reflecting the temporal constraints $S_j - S_i \geq d'_{ij}$ that are implied by the original temporal constraints and the added disjunctive precedence constraints. For example, the modified distance matrix can be chosen to be equal to the elementwise minimal matrix D' with $d'_{gh} \geq \max(d_{gh}, \min_{i \in A_\mu} \max_{j \in B_\mu}(d'_{gi} + p_i + d'_{jh}))$ for all $g, h \in V$ and all $\mu = 1, \ldots, \nu$ which satisfies the triangle inequalities (1.6). For distances d'_{0h} we may choose $d'_{0h} = S_h$ $(h \in V)$, where S is the minimal point of search space $\mathcal{P}$ computed by Algorithm 3.2.

The question which consistency test should actually be used at which node has to be investigated with care. The reason for this is that intuitively there is a tradeoff between the efficiency (i.e., the computation time required) and the effectiveness (i.e., the decrease in size of the search space) of a test. As a rule, the deeper the enumeration node, the less time should be spent with consistency tests. In any case, the search space reduction algorithm (cf. Algorithm 1.5) should be implemented in the form of a label-correcting procedure iterating the hypothetical temporal constraints whose validity may be affected by the last constraint added (either an imposed disjunctive precedence constraint or a temporal constraint arising from applying a consistency test). In branch-and-bound algorithms for the project duration problem with renewable resource constraints, De Reyck and Herroelen (1998*a*) and Schwindt (1998*c*) have applied the disjunctive activities test to two-element forbidden sets (De Reyck and Herroelen used the test as a preprocessing technique at the root node). Dorndorf et al. (2000*a*) report on favorable results using the workload-based disjunctive activities test and the unit-interval capacity test in a time-oriented branch-and-bound procedure for the same problem (the latter algorithm is briefly sketched in Subsection 3.1.4). Dorndorf et al. have

also experimented with the activity interval and general interval consistency tests, but on their testbed (projects with 100 or 500 activities) the additional search space reduction has been, on the average, outweighed by the increase in computation time. Finally, Laborie (2003) has been able to improve upon the results obtained by Neumann and Schwindt (2002) for the project duration problem with cumulative resources by using the balance test.

Next, we turn to **lower bounds** on the minimum objective function value. Let S be the minimal point of search space $\mathcal{P}$ under study (possibly reduced by applying consistency tests). Obviously, $lb_0 = f(S)$ represents a lower bound on the objective function value $\min_{S' \in \mathcal{P} \cap \mathcal{S}} f(S')$ of a best feasible schedule in $\mathcal{P}$. Within a branch-and-bound algorithm for the project duration problem with renewable resources, Schwindt (1998a) has used two further lower bounds lb_1 and lb_2, respectively being based on disjunctive activities and energetic reasoning. We first deal with lower bound lb_1 (see also Klein and Scholl 1998). Let $\overline{d}' \in \mathbb{Z}_{\geq 0}$ with $S_{n+1} \leq \overline{d}' \leq \overline{d}$ denote some hypothetical upper bound on the project duration. Clearly, the latest start time $LS_i = -d_{i0}$ of activity i is, under the assumption of a project deadline $\overline{d}'$, less than or equal to $\overline{d}' - d_{i,n+1}$, and the earliest completion time $S_i + p_i$ of activity i is independent of $\overline{d}'$. Now let $\{i, j\}$ be a forbidden set such that $\overline{d}' - d_{j,n+1} < S_i + p_i$ and $\overline{d}' - d_{i,n+1} < S_j + p_j$. Then activities i and j must overlap in time, which is impossible due to their excessive joint resource requirements. Consequently, $\overline{d}' + 1$ is a lower bound on the shortest project duration of all schedules in the search space. Moreover, $\overline{d}'$ must be increased by $\min(S_i + p_i + d_{j,n+1}, S_j + p_j + d_{i,n+1})$ units of time to avoid the above contradiction. Thus, instead of performing a binary search in set $[S_{n+1}, \overline{d}] \cap \mathbb{Z}$, we may directly compute the smallest deadline $\overline{d}' = lb_1$ which cannot be disproved as

$$lb_1 = \max(S_{n+1}, \max_{\{i,j\} \in \mathcal{F}} \min(S_i + p_i + d_{j,n+1}, S_j + p_j + d_{i,n+1}))$$

For given two-element forbidden sets $\{i, j\}$, calculating smallest deadline $\overline{d}'$ requires $\mathcal{O}(n^2)$ time. By applying the profile test from Subsection 1.3.4 to the project termination event $n+1$, Neumann and Schwindt (2002) have obtained a similar lower bound on the minimum project duration of projects with cumulative resources. The algorithm iterates hypothetical upper bounds $\overline{d}'$, which may be refuted based on lower and upper approximations to the loading profiles.

Now recall the concept of lower bound $w_k(a, b)$ on the workload to be processed on renewable resource $k \in \mathcal{R}^\rho$ in interval $[a, b[$ (see equations (1.13) and (1.14)). By replacing the earliest completion time $EC_i = ES_i + p_i$ in (1.13) with $S_i + p_i$, we obtain a corresponding lower bound referring to the search space $\mathcal{P}$ rather than to set $\mathcal{S}_T$. In particular, $w_k(S_i, \overline{d})$ represents a lower bound on the workload for resource k that must be processed after the earliest start time S_i of activity i, which takes at least $\lceil w_k(S_i, \overline{d})/R_k \rceil$ units of time. By taking the maximum with respect to all real activities $i \in V^a$ and all renewable resources $k \in \mathcal{R}^\rho$, we obtain lower bound

$$lb_2 = \max_{i \in V^a}(S_i + \max_{k \in \mathcal{R}^\rho} \left\lceil \frac{w_k(S_i, \overline{d})}{R_k} \right\rceil)$$

on the minimum project duration. Computing value lb_2 can be done in $\mathcal{O}(|\mathcal{R}^\rho| n \log n)$ time.

We briefly touch upon further, more time-expensive lower bounds on the minimum duration of projects with renewable resources, which can be found in Heilmann and Schwindt (1997), Brucker and Knust (2003), and Möhring et al. (2003) and will be used for the performance analysis of exact and heuristic methods for the project duration problem in Subsection 3.1.4. The latter two lower bounds are also described in more detail in Neumann et al. (2003*b*), Subsect. 2.5.8.

Heilmann and Schwindt (1997) discuss several lower bounds based on disjunctive activities, energetic reasoning, and a relaxation of the resource-constrained project scheduling problem (1.8) leading to a preemptive one-machine problem with release dates d_{0i}^{min} and quarantine times $d_{i,n+1}^{min}$ ($i \in V^a$).

Similarly to lower bound lb_1, the lower bound on the minimum project duration devised by Brucker and Knust (2003) is based on falsifying hypothetical project deadlines $\overline{d}'$. For a given value of $\overline{d}'$, the procedure of testing the consistency of deadline $\overline{d}'$ constructs a linear program and tries to show that it is unsolvable. At first, several consistency tests are applied in order to tighten the time windows $[S_i, LS_i]$ of individual activities $i \in V^a$ (recall that minimal point S coincides with the earliest schedule in set $\mathcal{P}$). For each pair (t, t') of consecutive earliest start or latest completion times of activities $i \in V^a$, the set of all tentative active sets $\mathcal{A}$ for interval $[t, t'[$ is then computed, where $S_i < t'$ and $LC_i > t$ for all $i \in \mathcal{A}$ and $d_{ij} < p_i$ for all $i, j \in \mathcal{A}$. For each set $\mathcal{A}$, a continuous decision variable $y_{\mathcal{A}} \geq 0$ is introduced providing the time during which $\mathcal{A}$ is in progress in interval $[t, t'[$ (i.e., during which precisely the activities $i \in \mathcal{A}$ overlap in time). The project duration is then minimized subject to the constraints that first, each real activity i is carried out for p_i units of time in the different sets $\mathcal{A}$ and second, the total execution time of all sets $\mathcal{A}$ belonging to some pair (t, t') is less than or equal to interval length $t' - t$. The latter problem can be formulated as a linear program in decision variables $y_{\mathcal{A}}$ and corresponds to the relaxation of problem (1.8) where the temporal constraints are replaced with the weaker release dates $d_{0i}^{min} = S_i$ and deadlines $d_{0i}^{max} = LS_i$. Moreover, activities are allowed to be interrupted during their execution. Since the number of tentative active sets $\mathcal{A}$ grows exponentially in n, it is expedient to solve the linear program by *column-generation techniques* (see, e.g., Goldfarb and Todd 1989, Sect. 2.6). The basic idea is to consider only a restricted working set of decision variables that are generated when needed. Each time the linear program with the current working set of decision variables has been solved to optimality, new decision variables are added to the working set or it is shown that the current basic solution is optimal. For finding an improving decision variable $y_{\mathcal{A}}$

to be added to the working set, Brucker and Knust use a branch-and-bound algorithm enumerating binary incidence vectors for sets $\mathcal{A}$.

Möhring et al. (2003) use a formulation of problem (1.8) as a binary linear program with time-indexed binary variables x_{it}, which has been proposed by Pritsker et al. (1969) for the first time. Decision variable x_{it} equals one if activity i is started at time t and zero, otherwise. For (approximatively) solving the continuous relaxation of the latter binary program, Möhring et al. apply a standard subgradient method (cf. Held et al. 1974) to a Lagrangean relaxation of the latter linear program, which substitutes the resource constraints into a linear penalty function. For given multipliers, the Lagrangean relaxation can be solved efficiently by transforming the problem into a minimum-cut problem in a cyclic network with upper arc capacities, where each node stands for one decision variable x_{it} (the time complexity of this approach is studied in more detail in Möhring et al. 2001). The main advantage of this approach is that it can be used for each objective function f which can be written in the form $\sum_{i \in V} w_{it} x_{it}$, where $w_{it} \in \mathbb{Z}$ and variables x_{it} are used in the above meaning. In addition, the approach can straightforwardly be generalized to the case of cumulative resources (see Selle 1999).

3.1.4 Additional Notes and References

Algorithm 3.1 combines the enumeration schemes of the branch-and-bound algorithms by Schwindt (1998*a*) and Neumann and Schwindt (2002) for the project duration problems with renewable-resource and cumulative-resource constraints, respectively (see also Schwindt 1999). In this subsection we briefly present alternative solution procedures that have been proposed in literature and present the results of an experimental performance analysis of the algorithms. We only consider algorithms coping with general temporal constraints. For the special case where instead of minimum and maximum time lags between activities precedence constraints are prescribed, we refer to the survey papers by Herroelen et al. (1998), Brucker et al. (1999), Hartmann and Kolisch (2000), and Kolisch and Padman (2001) and the literature cited therein.

We first deal with exact procedures for the **project duration problem with renewable resources**. By using ordinary precedence constraints instead of disjunctive precedence constraints for breaking up forbidden active sets, we obtain the enumeration scheme of a branch-and-bound algorithm that has been devised by De Reyck and Herroelen (1998*a*). Accordingly, the enumeration nodes correspond to time-feasible relations ρ which arise from the union of *minimal delaying modes* $\{i\} \times B$. This enumeration scheme will be discussed in more detail in Subsection 3.2.1.

The earliest branch-and-bound algorithm for the project duration problem is due to Bartusch et al. (1988). Their approach differs from the algorithm by De Reyck and Herroelen in the forbidden sets considered in the course of the algorithm. The forbidden sets F broken up in the latter algorithm (and, likewise, in Algorithm 3.1) are always active sets $\mathcal{A}(S,t)$ belonging to the

minimal point S of the search space $\mathcal{S}_T(\rho)$. If there is no forbidden active set $\mathcal{A}(S,t)$ for S at any time $t \geq 0$, schedule S is feasible, and no further pairs (i,j) are added to ρ. As we have already noticed in Subsection 3.1.1, the feasibility of S does not necessarily imply the feasibility of relation ρ. The algorithm of Bartusch et al. first computes all minimal forbidden sets $F \in \mathcal{F}$ for which the temporal constraints allow the simultaneous processing of all activities $i \in F$. Similarly to the enumeration scheme of the algorithm by De Reyck and Herroelen, enumeration nodes correspond to relations ρ in set V^a. The child nodes ρ', however, now arise from branching, for given minimal forbidden set F, over all pairs (i,j) of activities $i,j \in F$ such that relation $\rho' := \rho \cup \{(i,j)\}$ breaks up F. Leaves of the enumeration tree are either feasible relations ρ or relations ρ for which no further minimal forbidden set can be broken up by any time-feasible relation $\rho' \supset \rho$.

By substituting the disjunctive precedence constraints (3.1) into release dates

$$d_{0j}^{min} = \min_{i \in A}(S_i + p_i) \quad (j \in B) \tag{3.2}$$

where the right-hand side is the smallest completion time of some activity $i \in A$ with respect to the schedule S under consideration, one obtains the enumeration scheme of the branch-and-bound algorithm by Fest et al. (1999). The main advantage of this approach is that given distance matrix D, minimizers S of the project duration on the search space can be calculated in $\mathcal{O}(|B|n)$ time. Furthermore, there exists a very simple and effective dominance criterion, which enables fathoming nodes by comparing corresponding release date vectors. The drawback of the release-date based enumeration scheme is that constraints (3.2) only temporarily establish a precedence relationship between sets A and B. Since in contrast to the case of disjunctive precedence constraints, the right-hand side of (3.2) is a constant, the resource conflict caused by forbidden set $F = A \cup B$ is not definitely settled and thus one and the same resource conflict may be resolved repeatedly along a path from the root to some leaf of the enumeration tree. Computational experience, however, indicates that this situation can often be avoided by discarding enumeration nodes which due to unnecessary idle times cannot lead to quasiactive schedules (total-idle-time dominance rule, cf. Fest et al. 1999).

All algorithms mentioned thus far are based on breaking up forbidden sets. The constraint propagation algorithm by Dorndorf et al. (2000*c*) branches over the binary decision whether to schedule a given activity $i \in V^a$ at its (current) earliest possible start time ES_i or delaying i by introducing a release date $d_{0i}^{min} \geq ES_i + 1$. The large size of the corresponding complete enumeration tree is significantly reduced by applying the disjunctive activities and unit-interval capacity consistency tests and exploiting specific properties of active schedules.

We proceed with heuristic procedures for the project duration problem with renewable resources. Franck (1999), Ch. 4, has proposed the following *priority-rule method*. Preliminary variants of this algorithm have been de-

vised by Neumann and Zhan (1995) and Brinkmann and Neumann (1996). A streamlined version of Franck's algorithm is described in Franck et al. (2001*b*). At first, a preprocessing step is performed by applying the disjunctive activities consistency test to two-element forbidden sets. To construct a feasible schedule, a *serial schedule-generation scheme* is used (cf. Kolisch 1996), which in each iteration schedules one eligible activity $j \in V^a$ by fixing its start time S_j. An activity j is called eligible if all of its predecessors $i \in Pred^{\prec}(j)$ with respect to strict order $\prec$ in set V^a have been scheduled, where $i \prec j$ if (1) $d_{ij} > 0$ or (2) $d_{ij} = 0$ and $d_{ji} < 0$. From the set of eligible activities, the activity to be scheduled next is chosen according to a priority rule. Let C denote the set of all activities already scheduled. The activity j selected is started at the earliest point in time $t \in [ES_j, LS_j]$, where $ES_j = \max[d_{0j}, \max_{i \in C}(S_i + d_{ij})]$ and $LS_j = \min[-d_{0j}, \min_{i \in C}(S_i - d_{ji})]$, such that in interval $[t, t+p_j[$ the joint resource requirements by j and the activities $i \in C$ do not exceed the resource capacities. Due to the presence of maximum time lags, it may happen that for a selected activity j there is no such point in time t. Let $t' := \min_{i \in C}(S_i - d_{ji})$ then denote the latest start time of j due to the (induced) maximum time lags between scheduled activities $i \in C$ and activity j. To resolve the deadlock, an *unscheduling step* is performed, which cancels the start times of all scheduled activities $i \in C$ with $S_i \geq t'$ and increases the earliest start time ES_i of all scheduled activities $i \in C$ with $S_i = t'$ by one unit of time. The procedure is terminated if a prescribed maximum number of unscheduling steps have been performed or if all activities $j \in V^a$ have been scheduled. The number of required unscheduling steps can be markedly decreased on the average if activities of strong components in project network N are scheduled directly one after another, where N does not contain backward arc $(n+1, 0)$ (recall that when minimizing the project duration, we may delete the deadline $\overline{d}$ on the project termination).

Based on this priority-rule method, Franck (1999), Ch. 6, has also developed a schedule-improvement procedure of type *parallel genetic algorithm* (see also Franck et al. 2001*b*), which is an adaptation of a genetic algorithm by Hartmann (1998) for the project duration problem without maximum time lags. The genetic algorithm works on several subpopulations of equal size, where each island evolves separately until after a given number of iterations, some individuals migrate from one subpopulation to another one. The individuals are represented by feasible activity lists (i.e., complete strict orders $\ll$ in set V^a extending strict order $\prec$), which are transformed into schedules by applying the serial schedule-generation scheme with strict order $\prec$ substituted into $\ll$. The initial subpopulations are created by randomly biasing priority rules and transforming the resulting priority values $\pi(i)$ of activities i in an activity list $\ll$ by putting $i \ll j$ if (1) $i \prec j$ or (2) $j \not\prec i$ and $\pi(i) < \pi(j)$. At each iteration, two individuals are selected for crossover in each subpopulation according to a double roulette-wheel selection. By applying a one-point and a two-point crossover operation to those two individuals two new activity lists are generated. With a certain probability, the new activity lists are then sub-

jected to mutation by interchanging the positions of two adjacent activities in the list. Subsequently, the two activity lists are decoded into schedules using the serial schedule-generation scheme. If in the course of the schedule generation a maximum number of unscheduling steps has been performed, the violation of maximum time lags is allowed, which means that the resulting schedule is not time-feasible. Based on the resulting schedules, the fitness of the activity lists is calculated as the sum of the project duration and a penalty term for time-infeasibility of the schedule. Eventually, the worst two individuals in the subpopulation are replaced with the two new activity lists, provided that the new activity lists have a better fitness. These steps are iterated until one of five stop criteria is met: all individuals have the same fitness, a lower bound on the shortest project duration has been attained, a prescribed number of schedules have been evaluated, a feasible schedule has not been found within a given number of iterations, or the best feasible schedule found has not been improved within a given number of iterations.

A variant of the enumeration scheme of De Reyck and Herroelen (1998*a*) has been used by Cesta et al. (2002) for a *multi-pass heuristic*, where relation $\{i\} \times B$ is replaced with a pair (i, j) such that the addition of (i, j) to relation ρ breaks up some selected minimal forbidden set F. Set F is chosen from a given number of sampled minimal forbidden sets $F' \in \mathcal{F}$ with $F' \subseteq \mathcal{A}(S, t)$ for some $t \geq 0$. F is one of the sampled minimal forbidden sets with minimum "temporal flexibility" in terms of total slack times TF_h with $h \in F$, and pair (i, j) is chosen such that the resulting temporal flexibility for set F is maximum. The addition of pairs (i, j) to ρ is repeated until $\mathcal{S}_T(\rho) = \emptyset$ or minimal point $S = \min \mathcal{S}_T(\rho)$ is a feasible schedule. Within the multi-pass procedure, the temporal flexibility used for selecting pairs (i, j) is randomly biased, and thus in general several different feasible schedules are generated.

We now turn to the results of an experimental performance analysis. All of the above algorithms for the project duration problem with renewable resources except the branch-and-bound algorithm of Bartusch et al. (1988) have been tested on a test set consisting of 1080 problem instances with 100 real activities and 5 renewable resources each. The instances have been generated randomly by using the project generator ProGen/max (see Schwindt 1998*b* and Kolisch et al. 1999). The construction of projects can be influenced by means of control parameters for the problem size, shape of the project network, activity durations, time lags, and resource constraints. From the 1080 instances, 1059 possess a feasible solution. For 785 instances, an optimal solution is known.

Table 3.1 shows, in historical order, the results obtained by the different procedures, where the computation times refer to a Pentium personal computer with 200 MHz clock pulse (to account for different hardware, we have linearly scaled the computation times for De Reyck and Herroelen's and Franck's algorithms according to the corresponding clock pulse ratio). The results for the branch-and-bound procedure of De Reyck and Herroelen (1998*a*) are given as quoted by Dorndorf et al. (2000*c*). "Schwindt (1998*a*)

Table 3.1. Performance of algorithms for the project duration problem with renewable resources

Algorithm	t_{cpu}	p_{opt}	p_{ins}	p_{nopt}	p_{unk}	Δ_{lb}
De Reyck and Herroelen (1998*a*)	3 s	54.8 %	1.4 %	42.5 %	1.1 %	n. a.
	30 s	56.4 %	1.4 %	41.1 %	1.1 %	n. a.
Schwindt (1998*a*) BB	3 s	58.0 %	1.9 %	40.1 %	0.0 %	7.5 %
	30 s	62.5 %	1.9 %	35.6 %	0.0 %	7.0 %
	100 s	63.4 %	1.9 %	34.7 %	0.0 %	6.9 %
Schwindt (1998*a*) FBS	28.1 s	59.4 %	1.9 %	38.7 %	0.0 %	6.4 %
Fest et al. (1999)	3 s	58.1 %	1.9 %	34.1 %	5.9 %	10.9 %
	30 s	69.4 %	1.9 %	28.7 %	0.0 %	7.7 %
	100 s	71.1 %	1.9 %	27.0 %	0.0 %	7.0 %
Franck (1999) PR	0.16 s	57.2 %	1.9 %	40.9 %	0.0 %	7.3 %
Franck (1999) GA	12.1 s	60.1 %	1.9 %	38.0 %	0.0 %	5.3 %
Dorndorf et al. (2000*c*)	3 s	66.2 %	1.9 %	31.6 %	0.3 %	5.2 %
	30 s	70.4 %	1.9 %	27.7 %	0.0 %	4.8 %
	100 s	71.7 %	1.9 %	26.4 %	0.0 %	4.6 %
Cesta et al. (2002)	100 s	63.2 %	1.9 %	34.9 %	0.0 %	7.3 %

BB" and "Schwindt (1998*a*) FBS" designate the branch-and-bound algorithm of Schwindt (1998*a*) and its truncation to a filtered beam search heuristic (see Franck et al. 2001*b*). "Franck (1999) PR" and "Franck (1999) GA" stand for the priority-rule method and genetic algorithm by Franck (1999). The priority-rule method is performed with 14 different priority rules and the best schedule is returned. For the branch-and-bound procedures, t_{cpu} denotes an imposed time limit after which the enumeration is stopped. For the heuristics, t_{cpu} is the mean computation time. p_{opt}, p_{ins}, p_{nopt}, and p_{unk} denote the percentages of instances for which respectively an optimal schedule is found and optimality is proven, insolvability is shown, a feasible schedule is found whose optimality cannot be shown, or the solvability status remains unknown. In addition, we provide the mean percentage deviation Δ_{lb} of the project duration found from a lower bound lb on the minimum project duration, which has been calculated using techniques described in Heilmann and Schwindt (1997), the lower bound of Möhring et al. (2003) based on Lagrangean relaxation, and the lower bound of Brucker and Knust (2003) using column generation (see Subsection 3.1.3). For the algorithm of De Reyck and Herroelen (1998*a*), the published mean deviations from lower bound are based on values different from lb and are thus not listed. The mean refers to the instances which have been solved to feasibility by the respective algorithm. For the heuristic methods, we say that optimality is proven if the project duration obtained equals lower bound lb

and insolvability is shown if the consistency tests included reduce the search space to void.

As far as the exact algorithms are concerned, the results suggest that the most recent of the branch-and-bound procedures (Dorndorf et al. 2000*c*) is also the algorithm which performs best with respect to all five evaluation criteria. The good performance of the constraint propagation algorithm is primarily due to a clever search strategy and the effectiveness of the consistency tests, which are applied at every enumeration node. In particular, the mean deviation Δ_{lb} from lower bound is significantly smaller than for all remaining algorithms and for almost three quarters of the instances, the enumeration is completed within a time limit of 100 seconds. It is worth mentioning that all algorithms compared, except De Reyck and Herroelen's branch-and-bound procedure, are able to identify all insolvable instances and to find a feasible schedule for each solvable instance. The comparison of the results obtained when varying the time limit of the branch-and-bound procedures, however, indicates that solving all of the remaining open instances would probably require a prohibitively large computation time.

The priority-rule method provides feasible schedules with an acceptable deviation from lower bound within a very short amount of time. If more computation time is available, the genetic algorithm may be used to improve the initial schedule calculated by the priority-rule method. The comparison with Dorndorf et al.'s algorithm stopped after three seconds, however, shows that the latter algorithm also outperforms the heuristics. In addition, the data for the filtered beam search version of the branch-and-bound procedure of Schwindt (1998*a*) suggest that even better results may be obtained by a truncated version of Dorndorf et al.'s algorithm.

We proceed with the **project duration problem with cumulative resources**. To the best of our knowledge, there are only two algorithms for solving this problem: the branch-and-bound procedure devised by Neumann and Schwindt (2002), which is based on the enumeration scheme given by Algorithm 3.1, and a branch-and-bound algorithm that has been proposed by Laborie (2003).

The enumeration scheme of the latter procedure picks two distinct events i, j with $d_{ij} < 0$ and $d_{ji} < 0$ in each iteration and branches over the binary decision whether or not i occurs before j (i.e., $S_i \leq S_j - 1$ or $S_i \geq S_j$). The selection of events i, j is based on the upper and lower bounds $\overline{r}_k^<(h)$, $\overline{r}_k^\leq(h)$, $\underline{r}_k^>(h)$, $\underline{r}_k^\leq(h)$ on the inventory levels in resources $k \in \mathcal{R}^\gamma$ just before and at the occurrence, respectively, of events $h \in V^e$ (see Subsection 1.3.4). At each node of the enumeration tree, the balance test is used to reduce the time windows $[ES_h, LS_h]$ of events $h \in V^e$.

Table 3.2 shows the results of an experimental performance analysis of Neumann and Schwindt's and Laborie's algorithms. The test set has again been generated by ProGen/max and contains 360 instances with 10, 20, 50, or 100 events and 5 cumulative resources each. The computations have been

performed on a Pentium personal computer with 200 MHz clock pulse. For each instance we have imposed a time limit of 100 seconds. The mean deviation from lower bound Δ_{lb} is based on the lower bound lb_1 that is obtained by applying the profile test to the project termination event $n+1$ (see Subsection 3.1.3).

Table 3.2. Performance of algorithms for the project duration problem with cumulative resources

Algorithm	n	p_{opt}	p_{ins}	p_{noopt}	p_{unk}	Δ_{lb}
Neumann and Schwindt (2002)	10	66.7%	33.3%	0.0%	0.0%	0.3%
	20	48.9%	51.1%	0.0%	0.0%	2.2%
	50	51.1%	45.6%	1.1%	2.2%	1.3%
	100	55.6%	34.4%	7.8%	2.2%	1.2%
Laborie (2003)	50	53.3%	46.7%	0.0%	0.0%	1.5%
	100	63.4%	36.6%	0.0%	0.0%	0.9%

We first discuss the results obtained with the algorithm by Neumann and Schwindt (2002). For all 180 instances with 10 and 20 events, the enumeration is completed within the time limit. Even for the projects with 100 events, 90% of the instances can be either solved to optimality or proved to be insolvable. Put into perspective with the data displayed in Table 3.1, those results may indicate that problems with cumulative-resource constraints are more tractable than problems with renewable resources. As far as the computation of feasible schedules is concerned, the picture is different. There exist projects with 50 events for which after 100 seconds neither a feasible schedule can be found nor insolvability can be shown. With the branch-and-bound algorithm by Laborie (2003), however, the twelve open instances with 50 or 100 activities can be solved within less than 100 seconds (56 seconds on a HP 9000/785 workstation), which again confirms the benefit of efficient and effective consistency tests. The results for the projects with 10 or 20 activities are the same as for the algorithm by Neumann and Schwindt (2002).

3.2 Convexifiable Objective Functions

For convexifiable objective functions, time-constrained project scheduling with disjunctive precedence constraints can no longer be performed efficiently, and thus resource conflicts are settled by introducing ordinary precedence constraints. After the treatment of an enumeration scheme for generating candidate schedules, we discuss two alternative approaches to solving the relaxations: the primal approach, which will be used to solve the time-constrained project scheduling problem at the root node of the enumeration tree, and the dual approach for adding precedence constraints between activities of the

project. Whereas the primal steepest descent algorithm iterates over time-feasible schedules, the dual flattest ascent algorithm consecutively enforces the precedence constraints. Both algorithms are used within a branch-and-bound algorithm for minimizing convexifiable objective functions. In addition, we provide an overview of alternative solution procedures that have been devised in literature for specific convexifiable objective functions and discuss the results of an experimental performance analysis of the methods treated.

3.2.1 Enumeration Scheme

Precursors of the enumeration scheme to be discussed in this subsection have been proposed, independently, by Icmeli and Erengüç (1996) for the net present value problem with renewable resources and by De Reyck and Herroelen (1998*a*) for the project duration problem with renewable resources. Icmeli and Erengüç (1996) have considered the case of precedence constraints among activities instead of general temporal constraints. The enumeration scheme has arisen from the combination of the relaxation-based approach by Bell and Park (1990) and the concept of minimal delaying alternatives introduced by Demeulemeester and Herroelen (1992). Later on, Schwindt (2000*c*) has used the enumeration scheme within a branch-and-bound algorithm for the total earliness-tardiness cost problem with renewable resources. For solving the capital-rationed net present value problem, Schwindt (2000*a*) has expanded the enumeration scheme to cope with cumulative resources.

The algorithm mainly differs from the enumeration scheme considered in Subsection 3.1.1 in that forbidden active sets are broken up by ordinary instead of disjunctive precedence constraints. Hence, each enumeration node is associated with a relation ρ rather than with a set P of relations, and the search spaces $\mathcal{P}$ represent relation polytopes $S_T(\rho)$. The relations ρ arise from the union of minimal delaying modes $\{i\} \times B$, where B is a minimal delaying alternative for some forbidden set F and $i \in A$ with $A \subseteq V \setminus B$ being an appropriate set of activities to be chosen depending on the type of the underlying resource conflict. Accordingly, we obtain one enumeration node for each combination of activity $i \in A$ and minimal delaying alternative B. The relaxation to be solved at an enumeration node belonging to relation ρ consists in finding a (local) minimizer of objective function f on search space $S_T(\rho)$. In contrast to the case of regular objective functions, it can easily be verified whether or not the search space becomes void when passing from ρ to a child node's relation $\rho' = \rho \cup (\{i\} \times B)$ by checking the condition $d^{\rho}_{ji} + p_i \leq 0$ for each $j \in B$ (see Proposition 1.9). Updating the distance matrix $D(\rho)$ after the addition of pairs (i, j) with $j \in B$ can be achieved in $\mathcal{O}(n^2)$ time by using Algorithm 1.3 and observing that $\max_{j \in B}(d_{gi} + \delta_{ij} + d_{jh}) = d_{gi} + p_i + \max_{j \in B} d_{jh}$ for all $g, h \in V$.

The enumeration scheme is now as follows (cf. Algorithm 3.3). Q is a list of relations in set V and $\mathcal{C}$ again denotes the set of candidate schedules to be generated. At first, we put the empty relation $\rho = \emptyset$ on list Q and set $\mathcal{C} := \emptyset$.

We then check whether there is a cycle of positive length in project network N, in which case we return the empty set of candidate schedules. At each iteration we take some relation ρ from list Q and determine a minimizer S of f on relation polytope $\mathcal{S}_T(\rho)$. If schedule S is resource-feasible, we have found a candidate schedule, which is added to set $\mathcal{C}$. Otherwise, we scan S for a start time $t = S_i$ of some activity $i \in V$ such that active set $\mathcal{A}(S,t)$ includes a forbidden set F and compute the corresponding set $\mathcal{B}$ of all minimal delaying alternatives. For each minimal delaying alternative $B \in \mathcal{B}$ and each activity i from the respective set A we obtain one minimal delaying mode $\{i\} \times B$, which is joined with relation ρ and gives rise to the extension ρ' of ρ. If relation polytope $\mathcal{S}_T(\rho')$ is nonempty, ρ' is added to the list Q of unexplored enumeration nodes. We then take the next relation ρ from list Q and proceed in the same way until no more relations ρ remain in list Q and the set $\mathcal{C}$ of all candidate schedules is returned.

Algorithm 3.3. Enumeration scheme for convexifiable objective functions

Input: A project, convexifiable objective function f.
Output: Set $\mathcal{C}$ of candidate schedules.

initialize list of relations $Q := \{\emptyset\}$ and set of candidate schedules $\mathcal{C} := \emptyset$;
if $\mathcal{S}_T = \emptyset$ **then return** $\mathcal{C}$; (* cycle of positive length in N *)
repeat
  delete some relation ρ from list Q;
  determine minimizer S of f on $\mathcal{S}_T(\rho)$;
  if S is resource-feasible **then** $\mathcal{C} := \mathcal{C} \cup \{S\}$; (* candidate schedule found *)
  else (* introduce ordinary precedence constraints *)
    determine time t such that resource constraints (1.7) or (1.20) are violated for some $k \in \mathcal{R}^p \cup \mathcal{R}^\tau$;
    if $k \in \mathcal{R}^p$ **then**
      set $F := \mathcal{A}(S,t) \cap V^a$;
      compute set $\mathcal{B}$ of all minimal delaying alternatives for F;
    else
      set $F := \mathcal{A}(S,t) \cap V^e$;
      compute set $\mathcal{B}$ of all minimal delaying alternatives for F and k;
    for all $B \in \mathcal{B}$ **do**
      if $k \in \mathcal{R}^p$ **then** set $A := F \setminus B$; **elsif** $B \subseteq V_k^{e+}$ **then** set $A := V_k^{e-} \setminus F$;
      else set $A := V_k^{e+} \setminus F$;
      for all $i \in A$ **do**
        set $\rho' := \rho \cup (\{i\} \times B)$;
        if $\mathcal{S}_T(\rho') \neq \emptyset$ **then** add ρ on list Q; (* search space is nonempty *)
until $Q = \emptyset$;
return $\mathcal{C}$;

3.2.2 Solving the Relaxations: The Primal Approach

The relaxation to be solved at each node of the enumeration tree generated by Algorithm 3.3 corresponds to a time-oriented scheduling problem of type (1.2) where $\mathcal{S}_T$ is substituted into some relation polytope $\mathcal{S}_T(\rho)$ and f is a convexifiable objective function. To simplify writing, we consider the relaxation at the root node, where $\rho = \emptyset$, i.e., the resource relaxation of problem (P).

Recall that if objective function $f : \mathcal{S}_T \to \mathbb{R}$ is convexifiable, there exists a C^1-diffeomorphism $\varphi : \mathcal{S}_T \to X$ such that composite function $\psi : X \to \mathbb{R}$ with $\psi(x) = (f \circ \varphi^{-1})(x)$ for all $x \in X$ is convex and the image $X = \varphi(\mathcal{S}_T)$ of $\mathcal{S}_T$ under φ is a convex set. The continuity of φ and the compactness of $\mathcal{S}_T$ imply that the domain X of ψ is compact as well. If for given convexifiable objective function f, a diffeomorphism φ satisfying the conditions of Definition 2.29 is known, the relaxation can be solved by computing a minimizer x of ψ on X and returning schedule $S = \varphi^{-1}(x)$. The existence of such a minimizer x is easily seen as follows. By definition of ψ, the lower-level set L_α^ψ of ψ for given $\alpha \in \mathbb{R}$ equals the image of lower-level set L_α^f of f under φ, which is closed because of the lower semicontinuity of f. Consequently, the continuity of φ provides the closedness of any lower-level set L_α^ψ of ψ, which means that ψ is lower semicontinuous as well. Since X is compact, ψ always assumes its minimum on X. A minimizer x of ψ on X can be determined by the ellipsoid method, whose time complexity is polynomial in the input length of function ψ and set X. We stress, however, that the latter time complexity is not necessarily polynomial in the input length of the original relaxation (1.2).

For two special cases, which cover most convexifiable objective functions occurring in practice, the relaxation can be solved more efficiently on the average. We first consider the case where f is piecewise affine, convex, and sum-separable in the nodes $i \in V$ and the arcs $(i,j) \in E$ of project network N, i.e., f can be written in the form

$$f(S) = \sum_{i \in V} f_i(S_i) + \sum_{(i,j) \in E} f_{ij}(S_j - S_i)$$

where functions $f_i : [ES_i, LS_i] \to \mathbb{R}$ $(i \in V)$ and $f_{ij} : [d_{ij}, -d_{ji}] \to \mathbb{R}$ $((i,j) \in E)$ are piecewise affine and convex. The problem of minimizing a sum-separable function on set $\mathcal{S}_T$ is known as the *optimal-potential problem* in literature (cf. e.g., Rockafellar 1998, Sect. 1J). It is well-known that the optimal-potential problem with piecewise affine and convex functions f_i and f_{ij} is dual to the *convex-cost flow problem*

$$\begin{aligned} \text{Minimize} \quad & \sum_{(i,j) \in E} f_{ij}^*(u_{ij}) + \sum_{i \in V} f_i^*\Big(\sum_{(j,i) \in E} u_{ji} - \sum_{(i,j) \in E} u_{ij} \Big) \\ \text{subject to} \quad & \underline{c}_i \le \sum_{(j,i) \in E} u_{ji} - \sum_{(i,j) \in E} u_{ij} \le \overline{c}_i \quad (i \in V : i \ne 0) \end{aligned}$$

where the functions f_i and f_i^* and the functions f_{ij} and f_{ij}^* are conjugate to each other (see Rockafellar 1998, Sect. 8G). Recall that a function ϕ^* with

effective domain X^* (i.e., $\phi^*(y) < \infty$ for all $y \in X^*$) is conjugate to a function $\phi : X \to \mathbb{R}$ if $\phi^*(y) = \sup_{x \in X}(y^\top x - \phi(x))$ for all $y \in X^*$. For given functions f_i and f_{ij}, the corresponding conjugate functions f_i^* and f_{ij}^* are piecewise affine as well. The functions f_i^* and f_{ij}^* (up to an additive constant) and the lower and upper bounds $\underline{c}_i$ and $\overline{c}_i$ on supplies at nodes $i \in V$ can be determined by reversing the roles of breakpoints and slopes in passing from functions f_i and f_{ij} to their respective conjugates f_i^* and f_{ij}^*. The additive constants arise from evaluating a convenient point on the characteristic curves of f_i and f_{ij} (see Rockafellar 1998, Example 3 in Sect. 8F). The characteristic curve Γ of a convex function of one variable is the set of all points $(x, y) \in \mathbb{R}^2$ such that y is between the left-hand and the right-hand derivative of the function at x. The convex-cost flow problem can be solved in $\mathcal{O}(mn^2 \log[\sum_{i \in V}(|\overline{c}_i| + |\underline{c}_i|)])$ time by a generalization of the capacity-scaling algorithm for the min-cost flow problem (see Ahuja et al. 1993, Sect. 14.5).

The subcase where $f_i(S_i) = w_i S_i$ for all $i \in V$ and $f_{ij}(S_j - S_i) = 0$ for all $(i, j) \in E$ leads to the following min-cost flow problem (cf. e.g., Russell 1970):

$$\begin{aligned} \text{Minimize} \quad & \sum_{(i,j) \in E} -\delta_{ij} u_{ij} \\ \text{subject to} \quad & \sum_{(i,j) \in E} u_{ij} - \sum_{(j,i) \in E} u_{ji} = \begin{cases} \sum_{j=1}^{n+1} w_j, & \text{if } i = 0 \\ -w_i, & \text{otherwise} \end{cases} \qquad (i \in V) \\ & u_{ij} \geq 0 \quad ((i, j) \in E) \end{aligned}$$

We now turn to the second special case, where convexifiable objective function f is assumed to be continuously differentiable or sum-separable in the nodes $i \in V$ of N. In that case, the relaxation is amenable to an efficient *primal steepest descent approach*, which has been used by Schwindt (2000*c*) for solving the time-constrained total earliness-tardiness cost problem and by Schwindt and Zimmermann (2001) for solving the time-constrained net present value problem. We first review some basic concepts required for what follows. For notational convenience, we assume that function f possesses a continuation $\bar{f}$ from an open set $C \subseteq \mathbb{R}^{n+2}$ to $\mathbb{R}$ which is differentiable at the boundary points of $\mathcal{S}_T$. The *directional derivative* of $\bar{f}$ at point $S \in \mathcal{S}_T$ in direction $z \in \mathbb{R}^{n+2}$ is defined to be

$$d\bar{f}|_S(z) := \lim_{\lambda \downarrow 0} \frac{\bar{f}(S + \lambda z) - \bar{f}(S)}{\lambda} \tag{3.3}$$

if the limit exists. Now recall that function $\psi = f \circ \varphi^{-1}$ is convex and thus is directionally differentiable in any direction at any interior point of its domain (cf. Shor 1998, Sect. 1.2). Since $f = \psi \circ \varphi$ is a composition of a C^1-function and a finite convex function, f is directionally differentiable in any direction at any interior point of its domain as well. The latter property implies that the limit in (3.3) always exists. The derivative $d\bar{f}|_S(z)$ in direction of the i-th unit vector $z = e_i$ coincides with the right-hand S_i-derivative $\partial^+ \bar{f} / \partial S_i(S)$ of

$\bar{f}$ at S, and the left-hand S_i-derivative $\partial^- \bar{f}/\partial S_i(S)$ of $\bar{f}$ at S equals $-d\bar{f}|_S(z)$ where $z = -e_i$. The vectors of right-hand and left-hand S_i-derivatives of $\bar{f}$ at S are denoted by $\nabla^+ \bar{f}(S)$ and $\nabla^- \bar{f}(S)$, respectively. For fixed schedule S, derivative $d\bar{f}|_S(z)$ is a positively homogeneous function g of z (i.e., $g(\alpha z) = \alpha g(z)$ for all $\alpha > 0$ and all $z \in \mathbb{R}^{n+2}$). Under our assumption that objective function f is continuously differentiable or sum-separable in $i \in V$, derivative $g(z) = d\bar{f}|_S(z)$ at point S in direction z takes the form

$$g(z) = \sum_{i \in V: z_i > 0} \frac{\partial^+ \bar{f}}{\partial S_i}(S) z_i + \sum_{i \in V: z_i < 0} \frac{\partial^- \bar{f}}{\partial S_i}(S) z_i$$

In particular, if f is continuously differentiable, then $g(z) = \nabla \bar{f}(S)^\top z$, where $\nabla \bar{f}(S)$ is the derivative of $\bar{f}$ at S. As we will see later on (see Lemma 3.4), $\partial^+ \bar{f}/\partial S_i(S) \geq \partial^- \bar{f}/\partial S_i(S)$ for all $i \in V$, which implies that $g(z) = \sum_{i \in V} \max(\partial^+ \bar{f}/\partial S_i(S) z_i, \partial^- \bar{f}/\partial S_i(S) z_i)$. Consequently, g is a convex and thus sublinear function.

A direction $z \in \mathbb{R}^{n+2}$ is called a *descent direction* at $S \in \mathcal{S}_T$ if $d\bar{f}|_S(z) < 0$. z is termed a *feasible direction* at S if for some $\varepsilon > 0$, $S + \delta z \in \mathcal{S}_T$ for all $0 < \delta < \varepsilon$. Due to the convexity of $\mathcal{S}_T$, the latter condition is equivalent to the existence of some $\varepsilon > 0$ with $S + \varepsilon z \in \mathcal{S}_T$. Now let for given schedule $S \in \mathcal{S}_T$, $E(S) := \{(i,j) \in E \mid S_j - S_i = \delta_{ij}\}$ denote the set of arcs $(i,j) \in E$ for which temporal constraint $S_j - S_i \geq \delta_{ij}$ is active at S. Then direction z is feasible at S precisely if $z_0 = 0$ and $z_j - z_i \geq 0$ for all $(i,j) \in E(S)$. A (normalized first-order) *steepest feasible descent direction* at S is a feasible descent direction z at S with $\|z\| \leq 1$ minimizing derivative $g(z) = d\bar{f}|_S(z)$, where $\|\cdot\|$ is some vector norm in $\mathbb{R}^{n+2}$.

Now recall that any local minimizer of f on $\mathcal{S}_T$ is a global minimizer as well (cf. Proposition 2.30d). Obviously, a schedule S can only be a local minimizer of f on $\mathcal{S}_T$ if there is no feasible descent direction at S. Thus, any local minimizer S of f on $\mathcal{S}_T$ must satisfy the following necessary optimality condition (defining an *inf-stationary point*, see Kiwiel 1986):

$$\inf\{g(z) \mid z_0 = 0 \text{ and } z_j - z_i \geq 0 \text{ for all } (i,j) \in E(S)\} \geq 0 \qquad (3.4)$$

Condition (3.4) is sufficient for S to be a local minimizer of f on $\mathcal{S}_T$ if f is convex or if f is differentiable and $\nabla \bar{f}(S) \neq 0$. The objective function of the net present value problem is an example of a convexifiable and differentiable objective function f for which $\nabla \bar{f}(S) \neq 0$ for all $S \in \mathcal{S}_T$.

A classical approach to computing local minimizers are so-called *steepest descent algorithms*, which construct a sequence $S^1, S^2, \ldots, S^\nu$ of iterates such that $f(S^{\mu+1}) < f(S^\mu)$ for all $\mu = 1, \ldots, \nu - 1$. Steepest descent algorithms belong to the class of *feasible direction methods* introduced by Zoutendijk (1960). Feasible direction methods offer an efficient way of solving nonlinear programming problems with linear inequality constraints (cf. e.g., Jacoby et al. 1972, Sect. 7.5, or Simmons 1975, Sect. 8.1), in particular if the directional derivatives are easily obtained. Iterations of steepest descent algorithms consist of

two main phases: the *direction-finding phase* and the *line-search phase* (see Hiriart-Urruty and Lemaréchal 1993, Sect. II.2). The direction-finding phase determines a steepest feasible descent direction z at the current iterate S or establishes that there is no feasible descent direction at S. Line search provides a feasible destination $S' = S + \sigma z$ with $f(S + \sigma z) < f(S)$. σ is termed the *stepsize*. Algorithm 3.4 specifies a generic (primal) steepest descent algorithm.

Algorithm 3.4. Primal steepest descent algorithm

Input: MPM project network $N = (V, E, \delta)$, objective function f.
Output: Local minimizer S of f on set $\mathcal{S}_T$.

```
determine some time-feasible schedule S, e.g., S := ES;
repeat
  determine normalized feasible direction z at S with minimum g(z); (* direction-
  finding phase *)
  if g(z) < 0 then (* z is a descent direction *)
    determine stepsize σ in N at S; (* line-search phase *)
    set S := S + σz;
until g(z) ≥ 0;
return S;
```

We now deal with the **direction-finding phase** in more detail. The problem of finding a normalized steepest feasible descent direction at schedule S reads as follows:

$$\left.\begin{array}{ll} \text{Minimize} & g(z) \\ \text{subject to} & z_j - z_i \geq 0 \quad ((i,j) \in E(S)) \\ & z_0 = 0 \\ & \|z\| \leq 1 \end{array}\right\} \tag{3.5}$$

The feasible region of problem (3.5) is compact and nonempty since $z = 0$ is always a feasible solution. The choice of vector norm $\|\cdot\|$ is of crucial importance for the efficiency of the steepest descent algorithm. For what follows, we assume that $\|\cdot\|$ is chosen to be supremum norm, i.e., $\|z\| = \|z\|_\infty := \max_{i \in V} |z_i|$, which means that normalization constraint $\|z\| \leq 1$ can be stated as $-1 \leq z_i \leq 1$ for all $i \in V$. In this case, all constraints of problem (3.5) are linear, and (3.5) can easily be transformed into a linear program by introducing an additional variable y_i for each $i \in V$ together with the constraints $y_i \geq \partial^+ \tilde{f}/\partial S_i(S) z_i$ and $y_i \geq \partial^- \tilde{f}/\partial S_i(S) z_i$ and replacing $g(z)$ with $\sum_{i \in V} y_i$.

In the following we consider a relaxation of problem (3.5) which can be solved in linear time. To this end, we again assume that $\|z\| = \|z\|_\infty$, but we only consider a subset of the temporal constraints that are active at S. The active temporal constraints to be taken into account are chosen such that the

corresponding rows of the coefficient matrix are linearly independent. As it is well-known from the theory of network flows, the directed graph $G = (V, E_G)$ whose arc set E_G contains the arcs belonging to the selected active temporal constraints represents a spanning forest of project network N (see, e.g., Ahuja et al. 1993, Sect. 11.2). Proposition 2.28b tells us that G can be chosen to be a spanning tree of N precisely if S is a vertex of $\mathcal{S}_T$. The *steepest descent problem* (SDP) at schedule S can now be formulated as follows:

$$\left.\begin{array}{lll} \text{Minimize} & g(z) & \\ \text{subject to} & z_j - z_i \geq 0 & ((i,j) \in E_G) \\ & z_0 = 0 & \\ & -1 \leq z_i \leq 1 & (i \in V) \end{array}\right\} \text{(SDP)}$$

A direction z solving steepest descent problem (SDP) is called an *optimal direction* at S. Of course, we have to pay a price for the efficiency with which (SDP) can be solved. At degenerate points S of $\mathcal{S}_T$, where $E_G \subset E(S)$, optimal directions may no longer be feasible directions at S. In the latter case, line search will provide the stepsize $\sigma = 0$, and the set of selected active constraints is modified without leaving the current iterate S. Since (SDP) is a relaxation of problem (3.5), schedule S satisfies the necessary optimality condition (3.4) if $z = 0$ is an optimal direction at S.

We show how for a given schedule $S \in \mathcal{S}_T$ the steepest descent problem can be solved in linear time. The procedure is based on two fundamental properties of problem (SDP). First, it always possesses an integral solution and second, it can be decomposed into two independent subproblems with linear objective functions.

Proposition 3.3. *Let f be a differentiable or sum-separable convexifiable objective function. Then there is an integer-valued solution z to* (SDP).

Proof. If f is differentiable or sum-separable, then objective function $g(z) = \sum_{i \in V: z_i > 0} \partial^+ \bar{f}/\partial S_i(S) z_i + \sum_{i \in V: z_i < 0} \partial^- \bar{f}/\partial S_i(S) z_i$. It follows that g is linear on each octant and continuous. Since $z = 0$ is a feasible solution to (SDP), the continuity of g implies that (SDP) is solvable. In addition, the coefficient matrix of (SDP) is totally unimodular, which means that a feasible solution z minimizing g on a given octant can always be chosen to be integral. □

We proceed with the decomposition of the steepest descent problem (SDP) into two independent subproblems where we only consider nonnegative directions $z \geq 0$ or nonpositive directions $z \leq 0$ and which are respectively denoted by (SDP$^+$) and (SDP$^-$). For (SDP$^+$) objective function $g(z)$ equals $\nabla^+ \bar{f}(S)^\top z$, and for (SDP$^-$) we have $g(z) = \nabla^- \bar{f}(S)^\top z$.

$$\left.\begin{array}{lll} \text{Minimize} & g(z) = \nabla^+ \bar{f}(S)^\top z \parallel \nabla^- \bar{f}(S)^\top z & \\ \text{subject to} & z_j - z_i \geq 0 & ((i,j) \in E_G) \\ & z_0 = 0 & \\ & 0 \leq z_i \leq 1 \parallel -1 \leq z_i \leq 0 & (i \in V) \end{array}\right\} \text{(SDP}^+\text{)} \parallel \text{(SDP}^-\text{)}$$

We first need two preliminary lemmas.

Lemma 3.4. *Let f be some convexifiable objective function and let S be a time-feasible schedule. Then*

$$\nabla^+ \bar{f}(S) \geq \nabla^- \bar{f}(S)$$

Proof. We only consider the case where S is an interior point of $\mathcal{S}_T$ since by assumption, $\bar{f}$ is differentiable at boundary points of $\mathcal{S}_T$. Let $\varphi : \mathcal{S}_T \to X$ be a C^1-diffeomorphism satisfying the conditions of Definition 2.29 and let $\psi = f \circ \varphi^{-1}$. Since φ is continuous, $x = \varphi(S)$ is an interior point of X. Let $\nabla\varphi(S)$ be the Jacobian matrix of φ at point S. For given direction $z \in \mathbb{R}^{n+2}$, applying the chain rule (see Shapiro 1990, Proposition 3.6 (ii) or Scholtes 1990, Theorem 3.1) then provides $d\bar{f}|_S(z) = d\psi|_x(y)$ where $y = \nabla\varphi(S)\, z$ (recall that φ is continuously differentiable and that ψ is finite-valued convex and thus continuously Bouligand-differentiable at interior point of its domain). We then have $-d\bar{f}|_S(-z) = -d\psi|_x(-y)$. The convexity of ψ implies that $-d\psi|_x(-y) \leq d\psi|_x(y)$ and thus $-d\bar{f}|_S(-z) \leq d\bar{f}|_S(z)$ (see Hiriart-Urruty and Lemaréchal 1993, Sect. VI.1). The assertion follows from $\frac{\partial^+ \bar{f}}{\partial S_i}(S) = d\bar{f}|_S(e_i)$ and $\frac{\partial^- \bar{f}}{\partial S_i}(S) = -d\bar{f}|_S(-e_i)$. □

Lemma 3.5. *z is a feasible solution to* (SDP) *if and only if* $\max(0, z)$ *and* $\min(0, z)$ *are feasible solutions to* (SDP).

Proof. Let $z^+ := \max(0, z)$ and $z^- := \min(0, z)$. Trivially, for any direction $z \in \mathbb{R}^{n+2}$ we have $z = z^+ + z^-$.
Sufficiency: Let H denote the coefficient matrix of constraints $z_j - z_i \geq 0$ $((i, j) \in E_G)$, which coincides with the negative arc-node incidence matrix of spanning forest G. If z^+ and z^- are feasible solutions to (SDP), then $Hz^+ \geq 0$ and $Hz^- \geq 0$, which implies that $Hz^+ + Hz^- = H(z^+ + z^-) = Hz \geq 0$. In addition, $z_0 = z_0^+ + z_0^- = 0$ and $z_i = z_i^+ + z_i^- \geq 0 - 1 = -1$ and $z_i = z_i^+ + z_i^- \leq 1 - 0 = 1$ for all $i \in V$.
Necessity: Let z and z' be two feasible solutions to (SDP). Then it follows from elementary calculus that $\max(z, z')$ and $\min(z, z')$ are feasible solutions to (SDP) as well. By choosing $z' = 0$ we obtain the feasibility of directions z^+ and z^-. □

Theorem 3.6. *Let z^+ be a solution to* (SDP^+) *and let z^- be a solution to* (SDP^-). *Then $z = z^+ + z^-$ solves problem* (SDP).

Proof. We first show that $g(z) = g(z^+) + g(z^-)$. As a consequence of Lemma 3.4 we have $g(z) = \sum_{i \in V} \max(\partial^+ \bar{f}/\partial S_i(S) z_i, \partial^- \bar{f}/\partial S_i(S) z_i)$, from which it follows that g is convex. The positive homogeneity of g then implies the sublinearity and thus the subadditivity of g. Hence, $g(z) = g(z^+ + z^-) \leq g(z^+) + g(z^-)$. Since $\max(0, z)$ is a feasible solution to (SDP^+) (see Lemma 3.5), we have $g(z^+) \leq g(\max(0, z))$. Symmetrically it holds that

$g(z^-) \leq g(\min(0,z))$. By definition $g(\max(0,z)) = \sum_{i\in V:z_i>0} \partial^+\bar{f}/\partial S_i(S)z_i = g(z) - \sum_{i\in V:z_i<0} \partial^-\bar{f}/\partial S_i(S)z_i$ and $g(\min(0,z)) = \sum_{i\in V:z_i<0} \partial^-\bar{f}/\partial S_i(S)z_i$. Thus, $g(z^+) \leq g(z) - g(\min(0,z)) \leq g(z) - g(z^-)$, i.e., $g(z) \geq g(z^+) + g(z^-)$.

Due to Lemma 3.5, problem (SDP) can now equivalently be stated as

$$\begin{array}{ll} \text{Minimize} & g(z') + g(z'') \\ \text{subject to} & z'_j - z'_i \geq 0,\ z''_j - z''_i \geq 0 \quad ((i,j) \in E_G) \\ & z'_0 = z''_0 = 0 \\ & 0 \leq z'_i \leq 1,\ -1 \leq z''_i \leq 0 \quad (i \in V) \end{array}$$

Since in the latter problem the vectors z' and z'' are unrelated, the problem decomposes into the two independent problems (SDP$^+$) and (SDP$^-$) with corresponding solutions z^+ and z^-. □

For solving problem (SDP$^+$) we make use of the following property of forests. A forest G with at least one node possesses a source i with at most one successor or a sink i with exactly one predecessor. We call such a node i an extremal node of G. Now let $c_i := \partial^+\bar{f}/\partial S_i(S)$ be the right-hand S_i-derivative of $\bar{f}$ at point S. If there is a source $i \neq 0$ of spanning forest G with $c_i \leq 0$, then there is a solution z^+ to (SDP$^+$) satisfying $z_i^+ = 0$. Conversely, if there is a sink $i \neq 0$ of G with $c_i > 0$, then $z_i^+ = 1$ for any solution to (SDP$^+$). In both cases node i (and all incident arcs) can be deleted from G. If there is no source i with $c_i \leq 0$ and no sink i with $c_i > 0$, then V necessarily contains a source i with at most one successor j (and $c_i > 0$) or a sink i with exactly one predecessor j (and $c_i \leq 0$). In both cases, i is delayed exactly if j is delayed, i.e., $z_i^+ = z_j^+$. Thus, nodes i and j can be coalesced into an aggregate activity with partial derivative $c_i + c_j$ (which corresponds to the directional derivative of $\bar{f}$ at S in direction z with $z_h^+ = 1$ for $h \in \{i,j\}$ and $z_h^+ = 0$, otherwise). We perform these steps until all nodes aside from 0 have been deleted from G.

Algorithm 3.5 provides an $\mathcal{O}(n)$-time implementation of the above procedure, where $Pred(i) := \{j \in V \mid (j,i) \in E_G\}$ and $Succ(i) := \{j \in V \mid (i,j) \in E_G\}$ denote the sets of immediate predecessors and successors of node i in G. To achieve the linear time complexity, we use an indices-representation of forests, which is similar to the data structure discussed in Ahuja et al. (1993), Sect. 11.3. We associate two indices $pred_i$ and $orient_i$ with each node $i \in V$. For each component C of G, we identify a specially designated node, called the root of C. If i is not a root node, $pred_i$ provides the predecessor of i in G on the unique (undirected) path from the root to i, and the orientation index $orient_i$ equals 1 if G contains arc $(pred_i, i)$ and -1, otherwise. For a root node i, we set $pred_i := -1$ and $orient_i := 0$. In addition, the nodes i of G are stored in some depth-first traversal order of G, starting in each component C at the root node. Then the last unvisited node $i \in U$ with respect to that order is always an extremal node of the subgraph G_U of G induced by set U. If $orient_i \leq 0$, i is a source of G_U, and if $orient_i = 1$, i is a sink of G_U. For $orient_i \neq 0$, the predecessor $j \in Pred(i)$ or successor $j \in Succ(i)$, respectively,

is given by $pred_i$. The sets $C(j)$ of coalesced nodes can efficiently be identified via a labelling technique.

Algorithm 3.5. Direction-finding phase

Input: Objective function f, schedule S, spanning forest G of project network N.
Output: Solution z^+ to (SDP$^+$).

set $U := V$, $z^+ := 0$, $c := \nabla^+ \bar{f}(S)$, and $C(i) := \{i\}$ for all $i \in V$;
while $U \neq \{0\}$ **do**
  if U contains a node $i \neq 0$ with $Pred(i) \cap U = \emptyset$ and $|Succ(i) \cap U| \leq 1$ **then**
    set $U := U \setminus \{i\}$;
    if $c_i > 0$ and $Succ(i) \cap U = \{j\}$ **then** set $c_j := c_j + c_i$ and $C(j) := C(j) \cup C(i)$;
  else
    determine a node $i \in U$, $i \neq 0$ with $Succ(i) \cap U = \emptyset$ and $Pred(i) \cap U = \{j\}$;
    set $U := U \setminus \{i\}$;
9: **if** $c_i > 0$ **then** set $z_h^+ := 1$ for all $h \in C(i)$;
  else set $c_j := c_j + c_i$ and $C(j) := C(j) \cup C(i)$;
return z^+;

The mirror problem (SDP$^-$) can be solved by a similar procedure where z^+ is replaced with z^-, vector c is initialized with the left-hand derivative $\nabla^- \bar{f}(S)$ at schedule S, the roles of predecessors and successors in G are reversed, and z_h^- is put to -1 on line 9. Theorem 3.6 says that $z = z^+ + z^-$ is an optimal direction at S.

In general, the **line-search phase** at schedule S is performed by computing an optimal stepsize $\sigma \geq 0$ such that destination schedule $S' = S + \sigma z \in \mathcal{S}_T$ minimizes f on the line segment ℓ in $\mathcal{S}_T$ passing through S in direction z. In certain cases, however, it is more efficient to proceed with a suboptimal descent step (see Jacoby et al. 1972, Sect. 5.1) because first, finding an optimal stepsize is expensive or second, moving to a minimizer S' on line segment ℓ may cause a zigzagging phenomenon. Schwindt (2000*c*) and Schwindt and Zimmermann (2001) have used the following stepsize σ in their steepest descent algorithms for the total earliness-tardiness cost and the net present value problems. Each activity $i \in V$ with $z_i \neq 0$ can at most be shifted until some temporal constraint $S_j - S_i \geq \delta_{ij}$ with $(i,j) \notin E_G$ becomes active, i.e.,

$$\sigma \leq \sigma_1(i) := \min_{(i,j) \in E : z_i > z_j} \frac{S_j - S_i - \delta_{ij}}{z_i - z_j}$$

$\sigma_1(i)$ may be equal to 0 if S is a degenerate point of $\mathcal{S}_T$. If f is not binary-monotone (see Subsection 2.3.1), we stop shifting i when crossing a kink of $\bar{f}$, i.e.

$$\sigma \leq \sigma_2(i) := \min\{\sigma' > 0 \mid \frac{\partial^- \bar{f}}{\partial S_i}(S + \sigma' z) < \frac{\partial^+ \bar{f}}{\partial S_i}(S + \sigma' z)\}$$

where for convenience we define $\min \emptyset := \infty$. Note that we have $\min_{i \in V} \sigma_2(i) = \min\{\sigma' > 0 \mid -d\bar{f}|_{S+\sigma' z}(-z) < d\bar{f}|_{S+\sigma' z}(z)\}$.

Accordingly, stepsize σ is chosen to be

$$\sigma = \min(\min_{i \in V} \sigma_1(i), \min_{i \in V} \sigma_2(i)) \tag{3.6}$$

where $\sigma_2(i) := \infty$ for all $i \in V$ if f is binary-monotone. For the general case of an objective function that is neither piecewise affine nor binary-monotone, we in addition have

$$\sigma \leq \min\{\sigma' > 0 \mid d\bar{f}|_{S+\sigma' z}(z) = 0\}$$

When moving from S to destination $S' = S + \sigma z$, spanning forest G is updated as follows. At first, we delete all arcs (g, h) from G for which $z_h > z_g$. If $\sigma = \sigma_1(i)$ for some $i \in V$, a new temporal constraint $S_j - S_i \geq \delta_{ij}$ becomes active and the corresponding arc (i, j) is added to G.

For the time-constrained net present value problem, Schwindt and Zimmermann (2001) have shown the following plausible statement, which readily carries over to the more general case of piecewise affine or binary-monotone objective functions f.

Proposition 3.7 (Schwindt and Zimmermann 2001). *If in Algorithm 3.3 the initial schedule is chosen to be the earliest schedule ES and the stepsizes σ are calculated according to (3.6), then at each iterate S there is a solution z to steepest descent problem* (SDP) *with $z \geq 0$.*

Under the assumptions of Proposition 3.7, it is thus sufficient to solve subproblem (SDP$^+$) for computing optimal directions z.

For piecewise affine or binary-monotone objective functions, the number of iterates needed to reach a schedule S satisfying necessary optimality condition (3.4) can markedly be decreased by using an acceleration technique. Consider the spanning forest G arising from deleting all arcs (i, j) with $z_j > z_i$ and let i be an activity with $\sigma = \min(\sigma_1(i), \sigma_2(i))$. All components C of G consist of nodes j with identical z_j. If there is a component of G which does not contain node i and for whose nodes j we have $z_j \neq 0$, those nodes can be shifted further without recomputing a new steepest descent direction. By shifting the components in order of nondecreasing minimum slacks between component nodes i and nodes j with $z_i > z_j$, we obtain the acceleration step displayed in Algorithm 3.6. If the one-sided S_i-derivatives of $\bar{f}$ are obtained in constant $\mathcal{O}(1)$ time, the algorithm can be implemented to run in $\mathcal{O}(m \log m)$ time by maintaining a Fibonacci heap of arcs $(i, j) \in E$ with $z_i > z_j$ and a Fibonacci heap of nodes $i \in V$ with $z_i \neq 0$ that are sorted according to nondecreasing slack times $\frac{S_j - S_i - \delta_{ij}}{z_i - z_j}$ and $\sigma_2(i)$, respectively.

We finally notice that if f is binary-monotone and $z \geq 0$, the resulting destination schedule S is always a vertex. Furthermore, it can be shown that in case of regular and in case of so-called *antiregular* objective functions f, which are componentwise nonincreasing in start times S_i, the steepest descent algorithm with the acceleration step included reaches the respective minimizers ES and LS after one iteration, independently of the initial schedule chosen (see Schwindt and Zimmermann 2001).

Algorithm 3.6. Acceleration step

Input: MPM project network $N = (V, E, \delta)$, schedule S, direction z, spanning forest G of project network N.
Output: Destination schedule S, updated spanning forest G.

for all $(i,j) \in E_G$ with $z_j > z_i$ **do** set $E_G := E_G \setminus \{(i,j)\}$;
while $z \neq 0$ **do**
  determine a node $i \in V$ with $z_i \neq 0$ and minimum slack $\sigma = \min(\sigma_1(i), \sigma_2(i))$;
  if $\sigma = \sigma_1(i)$ **then** (* update spanning forest G *)
    set $E_G := E_G \cup \{(i,j)\}$ for some arc $(i,j) \in E$ with $z_i > z_j$ and $\frac{S_j - S_i - \delta_{ij}}{z_i - z_j} = \sigma(i)$;
  determine node set $C(h)$ of component with $i \in C(h)$;
  for all $g \in C(h)$ **do** set $S_g := S_g + \sigma$ and $z_g := 0$;
return schedule S;

3.2.3 Solving the Relaxations: The Dual Approach

Let ρ be a relation in activity set V to be extended by a minimal delaying mode $\{i\} \times B$ in the course of Algorithm 3.3. For computing a minimizer on the reduced search space $\mathcal{S}_T(\rho')$ of the resulting relation $\rho' = \rho \cup \{\{i\} \times B\}$, it is often more expedient to use a dual approach rather than re-performing the primal steepest descent algorithm from scratch. The basic principle of the *dual flattest ascent approach* is to start with the minimizer S of f on $\mathcal{S}_T(\rho)$ and to perform an outer approximation towards set $\mathcal{S}_T(\rho')$, where the distance to $\mathcal{S}_T(\rho')$ is stepwise decreased at locally minimal cost. More precisely, at each iteration we consider moving in feasible directions z such that first, $z_j - z_i \geq 1$ for all $j \in B$ and second, the directional derivative $g(z)$ at iterate S is minimum. We refer to such a direction as a *flattest feasible ascent direction* at S. Let $\Delta(S, \mathcal{S}_T(\rho')) := \inf_{S' \in \mathcal{S}_T(\rho')} \|S' - S\|_\infty = \max_{j \in B}(S_i + p_i - S_j)^+ = (S_i + p_i - \min_{j \in B} S_j)^+$ denote the distance between S and set $\mathcal{S}_T(\rho')$. The first condition ensures that $\Delta(S + \sigma z, \mathcal{S}_T(\rho')) < \Delta(S, \mathcal{S}_T(\rho'))$ provided that stepsize $\sigma > 0$, whereas the second requirement means that the first-order approximation of the increase in the objective function value when moving from S to $S + \sigma z$ is minimum. We notice that if f is not a convex and piecewise affine function, this increase may also be negative, and thus in the general case we have to consider *normalized* flattest ascent directions z at S.

Algorithm 3.7 shows a generic flattest ascent algorithm, where for simplicity we assume that $\rho = \emptyset$ and $\mathcal{S}_T(\rho') \neq \emptyset$. At each iteration of the algorithm we first remove those activities j from minimal delaying alternative B for which precedence constraint $S_j \geq S_i + p_i$ has already been enforced. The arcs (i,j) corresponding to the latter precedence constraints are added to project network N in order to ensure that they are observed at all subsequent iterations. Next, we compute a normalized flattest ascent direction z at S. If $B = \emptyset$ and $g(z) = 0$, we have reached set $\mathcal{S}_T(\rho')$ and there is no feasible

descent direction z at S. Otherwise, we determine an appropriate stepsize σ, move to destination schedule $S + \sigma z$, and put $S := S + \sigma z$.

Algorithm 3.7. Dual flattest ascent algorithm

Input: MPM project network $N = (V, E, \delta)$, objective function f, time-optimal schedule S, minimal delaying mode $\{i\} \times B$.
Output: Local minimizer S of f on set $\mathcal{S}_T(\{i\} \times B)$.

repeat
  for all $j \in B$ with $S_j \geq S_i + p_i$ **do**
    remove j from set B and add arc (i, j) with weight $\delta_{ij} := p_i$ to N;
  determine normalized flattest feasible ascent direction z at S; (* direction-finding phase *)
  if $B \neq \emptyset$ or $g(z) < 0$ **then**
    determine stepsize σ in N at S; (* line-search phase *)
    set $S := S + \sigma z$;
until $B = \emptyset$ and $g(z) = 0$;
return S;

In what follows, we study the direction-finding and line-search phases in more detail. During the **direction-finding phase** of the algorithm we have to determine a flattest feasible ascent direction z at the given iterate S. In analogy to (3.5), the latter problem can be formulated as follows, where $\{i\} \times B$ is the minimal delaying mode under consideration:

$$\left.\begin{array}{lll} \text{Minimize} & g(z) & \\ \text{subject to} & z_h - z_g \geq 0 & ((g,h) \in E(S)) \\ & z_0 = 0 & \\ & z_j - z_i \geq 1 & (j \in B) \\ & \|z\| \leq 1 & \end{array}\right\} \tag{3.7}$$

The normalization constraint $\|z\| \leq 1$ may be deleted if f is convex and piecewise affine (the objective functions of the total inventory holding cost and total earliness-tardiness cost problems are examples of such an objective function). We notice that in contrast to the steepest descent problem (3.5), problem (3.7) does not necessarily possess a feasible solution. It is easily seen, however, that under the assumption that relation ρ' is time-feasible, i.e., $\mathcal{S}_T(\rho') \neq \emptyset$, there is always a flattest ascent direction at S.

In analogy to the primal steepest descent algorithm treated in Subsection 3.2.2, we choose the vector norm $\|\cdot\|$ in (3.7) to be the supremum norm and relax the problem by replacing the arc set $E(S)$ belonging to all active constraints at S with the arc set $E_G \subseteq E(S)$ of some spanning forest G of project network N. The resulting problem will be referred to as the *flattest ascent direction problem* (FAP) at S.

$$\left.\begin{array}{lll} \text{Minimize} & g(z) & \\ \text{subject to} & z_h - z_g \geq 0 & ((g,h) \in E_G) \\ & z_0 = 0 & \\ & z_j - z_i \geq 1 & (j \in B) \\ & -1 \leq z_h \leq 1 & (h \in V) \end{array}\right\} \text{(FAP)}$$

A solution z to (FAP) is again called an *optimal direction* at S. Our approach to solving the flattest ascent direction problem is based on a decomposition of the problem into two subproblems, where we respectively enforce all activities $j \in B$ to be right-shifted (i.e., $z_j = 1$) or activity i to be left-shifted (i.e., $z_i = -1$). The problems where in (FAP) we replace $z_j - z_i \geq 1$ $(j \in B)$ by the corresponding constraints $z_i = 0$ and $z_j = 1$ $(j \in B)$ or $z_i = -1$ and $z_j \geq 0$ $(j \in B)$ are denoted by (FAP$^+$) or (FAP$^-$), respectively.

Proposition 3.8. *Flattest ascent problem* (FAP) *is unsolvable if and only if both problems* (FAP$^+$) *and* (FAP$^-$) *are unsolvable. If* (FAP) *is solvable, it is solved by any solution* z^+ *to* (FAP$^+$) *or by any solution* z^- *to* (FAP$^-$).

Proof. Analogously to the proof of Proposition 3.3 it can again be shown that, if (FAP) is solvable, there exists an integral solution z to (FAP). In the latter case, z_i may assume the two values 0 and -1. If $z_i = 0$, we have $z_j = 1$ for all $j \in B$. For $z_i = -1$, the constraints $z_j - z_i \geq 1$ $(j \in B)$ turn into $z_j \geq 0$ $(j \in B)$. □

As a consequence of Proposition 3.8, an optimal direction z at S can be computed by solving both subproblems (FAP$^+$) and (FAP$^-$) and choosing $z = z^+$ if $g(z^+) \leq g(z^-)$ and $z = z^-$, otherwise (where we write $g(z^+) = \infty$ or $g(z^-) = \infty$ if the respective subproblem is unsolvable). Like the steepest descent problem (SDP), problem (FAP$^+$) can be solved by using Algorithm 3.5 for (SDP$^+$) and its analogue for the mirror problem (SDP$^-$). To this end, we put $c_i := \infty$ and $c_j := -\infty$ for all $j \in B$ when we apply Algorithm 3.5, and we put $c_i := -\infty$ and $c_j := \infty$ for all $j \in B$ when using the algorithm for the mirror problem. Problem (FAP$^-$) can be dealt with analogously. In sum, computing an optimal direction z at S necessitates four calls to the direction-finding algorithms from Subsection 3.2.2 and thus can again be achieved in linear time.

The following proposition shows that if at current iterate S moving in any feasible descent direction z at S would increase the distance between S and $\mathcal{S}_T(\rho')$, then (FAP) can be solved by only one application of Algorithm 3.5 and its adaptation for the mirror problem. It can easily be seen (cf. Schwindt 2000*c*) that the conditions of the proposition are satisfied at each iterate if f is convex and piecewise affine.

Proposition 3.9. *Let S be a time-feasible schedule and assume that for given minimal delaying mode $\{i\} \times B$, $z = 0$ solves the steepest descent problem*

(SDP) *at point* S *with additional constraints* $z_j - z_i \geq 0$ *for all* $j \in B$. *If* (FAP$^+$) *is solvable, it is solved by some direction* $z^+ \geq 0$, *and if* (FAP$^-$) *is solvable, it is solved by some direction* $z^- \leq 0$.

Proof. Let z' be an optimal solution to (FAP$^+$). Since $z^+ := \max(0, z')$ satisfies all constraints of problem (SDP) (compare the proof of Lemma 3.5) and $z_j^+ - z_i^+ = 1$ for all $j \in B$, z^+ is a feasible solution to (FAP)$^+$ as well, and thus from the optimality of z' it follows that $g(z^+) \geq g(z')$. Moreover, direction $z'' := \min(0, z')$ is a feasible solution to problem (SDP) with $z_j - z_i \geq 0$ for all $j \in B$. For z' we have $g(z') = \sum_{i \in V: z_i' > 0} \partial^+ \bar{f}/\partial S_i(S) z_i' + \sum_{i \in V: z_i' < 0} \partial^- \bar{f}/\partial S_i(S) z_i' = g(z^+) + g(z'')$. Since the optimal objective function value of problem (SDP) with $z_j - z_i \geq 0$ for all $j \in B$ equals 0, it holds that $g(z'') \geq 0$. We conclude that $g(z^+) = g(z') - g(z'') \leq g(z')$, which due to $g(z^+) \geq g(z')$ provides $g(z^+) = g(z')$. From the feasibility of direction z^+ then follows the assertion. The proof for problem (FAP$^-$) is analogous, where $z^- := \min(0, z')$ and $z'' := \max(0, z')$. □

For given optimal direction z, the **line-search phase** yields an appropriate stepsize $\sigma \geq 0$ such that

$$\sigma \leq \sigma_3(j) := S_i + p_i - S_j$$

for all $j \in B$. $\sigma_3(j)$ is the amount by which the time lag between the starts of activities i and j has to be increased for satisfying precedence constraint $S_j \geq S_i + p_i$. In addition, σ is chosen such that destination schedule $S' = S + \sigma z$ is time-feasible and we do not move beyond a kink of g, i.e.,

$$\sigma = \min(\min_{h \in V} \sigma_1(h), \min_{h \in V} \sigma_2(h), \min_{j \in B} \sigma_3(j))$$

3.2.4 Branch-and-Bound

By providing the enumeration scheme given by Algorithm 3.3 with a search strategy, consistency tests, and lower bounds, we obtain a branch-and-bound procedure for problem (P) with convexifiable objective function f. For the same reasons as in Subsection 3.1.3 it is generally expedient to store list Q of unexplored enumeration nodes in a stack, i.e., to perform a depth-first search. Since the consistency tests discussed in Subsections 1.2.4 and 1.3.4 do not refer to the objective function, we may again apply all those tests in principle. The effectiveness of a given test, however, among other things strongly depends on the particular objective function under consideration. As for the case of regular objective functions, the objective function value $f(S)$ of a minimizer S of f on some search space $\mathcal{S}_T(\rho)$ may again serve as a lower bound lb_0 on the objective function value of the best feasible schedule in $\mathcal{S}_T(\rho)$. Selle (1999) and Kimms (2001*b*) have used the technique devised by Möhring et al. (2003) based on Lagrangean relaxation of the resource constraints (see Subsection 3.1.3) to

compute lower bounds for the net present value and total earliness-tardiness cost problems, respectively, with renewable resources.

Sometimes relations ρ can be excluded from further consideration because they are dominated by other relations ρ' in the sense that either the absence of feasible schedules in $\mathcal{S}_T(\rho')$ excludes the existence of feasible schedules in $\mathcal{S}_T(\rho)$ or the minimum objective function value of the best feasible schedule in $\mathcal{S}_T(\rho')$ can be proved to be not greater than for the best feasible schedule in $\mathcal{S}_T(\rho)$. The simplest type of dominance between relations is given by the set inclusion of relation polytopes: relation ρ' dominates ρ if $\mathcal{S}_T(\rho) \subseteq \mathcal{S}_T(\rho')$. Since such *dominance rules* define a reflexive relation in the set of relations, one has to ensure by appropriate tie-breakers that "cross-pruning" (i.e., relation ρ' dominates relation ρ and vice versa) does not occur. The branch-and-bound algorithm may apply several dominance rules to newly generated relations ρ with corresponding minimal delaying mode $\{i\} \times B$.

The first dominance rule is as follows (cf. De Reyck and Herroelen 1998*a*). We add all activities $h \in \mathcal{A}(S,t) \setminus B$ with $d_{jh} \geq 0$ for some $j \in B$ to set B because they are delayed as well when shifting activities $j \in B$ behind the completion of activity i. If there is a minimal delaying alternative $B' \in \mathcal{B}$ with $B' \subset B$, relation ρ is dominated by relation ρ' belonging to minimal delaying mode $\{i\} \times B'$. The second dominance rule refers to a (possibly induced) minimum time lag between activity i and some activity i' of a delaying mode $\{i'\} \times B$ with the same minimal delaying alternative. If either (1) $d_{i'i} + p_i > p_{i'}$ or (2) $d_{i'i} + p_i = p_{i'}$ and (as tie-breaker) $i' < i$, then relation ρ can be fathomed because the completion time of activity i is greater than or equal to the completion time of activity i'.

Whereas the first two rules establish dominance between child nodes ρ of one and the same parent node, the following *subset-dominance rules* compare the recent child nodes ρ with (arbitrary) relations ρ' from which we have branched formerly or which remain on stack Q. The first subset-dominance rule has again been proposed by De Reyck and Herroelen (1998*a*). If the whole search space $\mathcal{S}_T(\rho')$ of a relation ρ' has been explored and if ρ' is a subset of ρ, relation ρ can be fathomed. This rule can be implemented to run quite efficiently by exploiting two properties of the enumeration tree (see Schwindt 1998*c*). First, $\rho' \subset \rho''$ for all descendants ρ'' of relations ρ' and second, in case of a depth-first search the parents ρ'' of relations ρ' with $\subseteq$-maximal completely explored search spaces $\mathcal{S}_T(\rho')$ are ancestors of ρ.

Neumann and Zimmermann (2002) have used a generalization of the latter rule in their branch-and-bound algorithm for the net present value problem with renewable resources. Comparing relations ρ and ρ' does not take into account the time lags that are induced by the distance matrix D. In other words, we may have $\mathcal{S}_T(\rho) \subseteq \mathcal{S}_T(\rho')$ though $\rho \not\supseteq \rho'$. Rather, condition $\mathcal{S}_T(\rho) \subseteq \mathcal{S}_T(\rho')$ can be checked by (elementwise) comparing the corresponding relation matrices $D(\rho)$ and $D(\rho')$, i.e., $\mathcal{S}_T(\rho) \subseteq \mathcal{S}_T(\rho')$ precisely if $d_{ij}^{\rho} \geq d_{ij}^{\rho'}$ for all $i,j \in V$.

The following subset-dominance rule by Schwindt (1998*c*) compares the recent child nodes ρ with relations ρ' on stack Q. If Q contains a relation $\rho' \subseteq \rho$ that is not an ancestor of ρ, then relation ρ can be deleted because $\mathcal{S}_T(\rho) \subseteq \mathcal{S}_T(\rho')$. This rule offers the advantage that no additional memory is required for storing enumeration nodes already visited. Of course, the rule can also be applied in a way to compare relation matrices rather than relations.

3.2.5 Additional Notes and References

In this subsection we briefly survey procedures for project scheduling with specific convexifiable objective functions and general temporal constraints. We first deal with primal algorithms for the time-constrained case, which may be used for solving the resource relaxation of problem (P). Kamburowski (1990) was probably the first who studied the **time-constrained net present value problem** with general minimum and maximum time lags between the start times of activities. He has proposed an adaptation of the approach by Grinold (1972) for ordinary precedence constraints to the case of general temporal constraints. Grinold's procedure is based on the transformation of the problem into a linear program by specifying a C^1-diffeomorphism φ which satisfies the conditions of Definition 2.29. Using specific properties of the linear program, the problem is solved by a vertex-following algorithm, the methods by Grinold (1972) and by Kamburowski (1990) differing in the pivot rule used. De Reyck and Herroelen (1998*b*) have generalized the recursive-search procedure by Herroelen et al. (1996) for the precedence-constrained net present value problem to the case of general temporal constraints. Starting at the earliest schedule, the activities of subtrees representing active temporal constraints and possessing a negative net present value are stepwise delayed in order to increase the net present value of the project. In contrast to all other procedures, the temporal constraints are represented by the distance matrix, i.e., their transitive closure, rather than by the project network. Neumann and Zimmermann (2000) have combined Kamburowski's procedure, equipped with a new pivot rule, and a preprocessing method proposed by Herroelen et al. (1996). The latter method delays all terminal activities with negative cash flows up to their latest start time (an activity is called terminal if it does not have successors in project network N aside from the project termination event $n+1$).

Table 3.3 compiles the results of an experimental performance analysis comparing the algorithms for the time-constrained net present value problem. The rows "Grinold (1972)" and "CPLEX" refer to the adaptation of Grinold's procedure to general temporal constraints with the original pivot rule and the primal simplex algorithm implemented in LP solver CPLEX 6.0 (among the different LP solvers available in the CPLEX package, the primal simplex method has shown the best performance). The performance of the algorithms has been evaluated on the basis of two test sets generated with ProGen/max. The test sets contain 1440 and 90 projects with 100 and 1000

activities, respectively (see Schwindt and Zimmermann 2001 for details). The results for the algorithm by De Reyck and Herroelen (1998*b*) are quoted from De Reyck (1998). We provide the mean number $\#it$ of iterations needed to reach an optimal solution (where "n.a." indicates that this number is not available) and the corresponding mean computation time t_{cpu} on an Intel 486 personal computer with 50 MHz clock pulse ($n = 100$) and a Pentium personal computer with 200 MHz clock pulse ($n = 1000$).

Table 3.3. Performance of primal algorithms for the time-constrained net present value problem

Algorithm	n	$\#it$	t_{cpu}
Grinold (1972)	100	22	26 ms
	1000	473	1.7 s
CPLEX	100	n.a.	570 ms
	1000	n.a.	118.6 s
Kamburowski (1990)	100	24	30 ms
	1000	577	2.5 s
De Reyck and Herroelen (1998*b*)	100	n.a.	831 ms
Neumann and Zimmermann (2000)	100	12	17 ms
	1000	219	1.0 s
Schwindt and Zimmermann (2001)	100	4	10 ms
	1000	17	0.6 s

The results depicted in Table 3.3 permit several conclusions. First, the methods based on Grinold's vertex-following algorithm show a much better performance than the primal simplex method applied to the linearized problem. Second, the preprocessing method allows to save roughly one half of the computation time. Third, the efficiency of the recursive-search method is poor, which is presumably less due to the recursion itself than rather to the use of the distance matrix, whose computation is expensive and which causes almost any vertex of set $\mathcal{S}_T$ to be degenerate. As a consequence, the algorithm performs many pivot steps that do not lead to a new vertex. Fourth, the steepest descent method appears as the most efficient solution procedure for the time-constrained net present value problem. If we reduce t_{cpu} by the time needed for computing the earliest schedule, the speed-up factor between the procedure of Neumann and Zimmermann (2000) and the steepest descent algorithm is more than six (cf. Schwindt and Zimmermann 2001). The small value for $\#it$ can be mainly attributed to the acceleration step, which for $n = 1000$ reduces the number of iterations by more than 90 %. This reduction does not lead to an equally large saving in computation time because the acceleration step is more time consuming than simple line search (recall

that the time complexity of the acceleration step is $\mathcal{O}(m \log m)$, whereas line search can be done in $\mathcal{O}(m)$ time).

Next, we consider the **time-constrained total earliness-tardiness cost problem**. The only algorithm for this problem we are aware of is the steepest descent procedure proposed by Schwindt (2000*c*). For the special case where only minimum time lags are present, Vanhoucke et al. (2001) have devised a recursive-search procedure, which is an adaptation of Herroelen et al.'s algorithm for the net present value problem. The time-constrained total earliness-tardiness cost problem can readily be transformed into a linear program by introducing two continuous variables $e_i \geq 0$ and $t_i \geq 0$ for each activity $i \in V$ along with the constraints $e_i \geq d_i - S_i - p_i$ and $t_i \geq S_i + p_i - d_i$. The objective function of the linear program then is $\sum_{i \in V}(w_i^e e_i + w_i^t t_i)$. Obviously, for $\mathcal{S}_T \neq \emptyset$ there is always an optimal solution satisfying $e_i = (d_i - S_i - p_i)^+$ and $t_i = (S_i + p_i - d_i)^+$ for all $i \in V$, i.e., e_i equals the earliness and t_i equals the tardiness of i. Notice that the existence of an equivalent linear program does not imply that the total earliness-tardiness cost is a linearizable objective function in the sense of Definition 2.29, which is obviously not true.

Table 3.4 compares the primal simplex algorithm with the steepest descent procedure. The analysis is based on two test sets with 100 and 1000 activities, respectively, containing 90 instances each (details are given in Neumann et al. 2003*b*, Sect. 3.5). The computations have been performed on a 200 MHz Pentium personal computer.

Table 3.4. Performance of primal algorithms for the time-constrained earliness-tardiness problem

Algorithm	n	$\#it$	t_{cpu}
CPLEX	100	367	539 ms
	1000	5035	58.3 s
Schwindt (2000*c*)	100	15	7 ms
	1000	139	3.8 s

The results are in line with those obtained for the net present value problem. Again, the steepest descent algorithm clearly outperforms the LP solver. However, the gap between both approaches is less important, which is due to two reasons. First, though the linear program contains more variables and constraints than for the net present value problem, the computation time decreases since the coefficient matrix of the constraints is now binary instead of real-valued. Second, since the objective function is no longer binary-monotone, the stepsizes for the steepest descent algorithm are typically much smaller, which is also indicated by the large increase in the number of iterations.

We proceed to the net present value and total earliness-tardiness cost problems with renewable or cumulative resources. We restrict ourselves to

procedures that are dedicated to the case of general temporal constraints between activities. For a review of various types of precedence-constrained net present value problems and solution procedures we refer to the survey paper by Herroelen et al. (1997). Algorithms for total earliness-tardiness cost problems with precedence constraints and renewable resources have been devised by Serafini and Speranza (1994*a*,*b*) and Vanhoncke et al. (2001). For solving the resource relaxation, Serafini and Speranza exploit the duality relationship between the latter problem and the convex-cost flow problem (see Subsection 3.2.2).

We first consider the **net present value problem with renewable resources**. The branch-and-bound algorithms by De Reyck and Herroelen (1998*b*) and Neumann and Zimmermann (2002) are both based on the enumeration scheme discussed in Subsection 3.2.1. The algorithms mainly differ in the procedures for solving the relaxations at the enumeration nodes. Whereas De Reyck and Herroelen (1998*b*) use their (primal) recursive-search method, Neumann and Zimmermann (2002) solve the initial resource relaxation at the root node by the primal steepest descent algorithm by Schwindt and Zimmermann (2001) and the relaxations at descendant nodes with a dual method resembling the flattest ascent algorithm dealt with in Subsection 3.2.3. In addition, De Reyck and Herroelen (1998*b*) and Neumann and Zimmermann (2002) have used disjunctive activities tests and dominance rules for reducing the size of the enumeration tree. Selle and Zimmermann (2003) have proposed a bidirectional priority-rule method for approximatively solving large-scale net present value problems. Similarly to the heuristic by Franck (1999) for the project duration problem (see Subsection 3.1.4), one activity is scheduled per iteration, where the essential difference is that certain activities, namely those with negative cash flows, are started at their *latest* feasible start time. An analysis of this schedule-generation scheme in Section 4.1 will show that the schedules obtained in this way are stable, provided that no unscheduling step is performed. Since the set of all optimal schedules may not contain a stable schedule, the heuristic may systematically miss the optimal solution. A similar result is known for the minimization of regular objective functions, where the parallel schedule-generation scheme yields nondelay schedules (see Kolisch 1996), among which there is not necessarily an optimal schedule.

Table 3.5 shows the results of an experimental performance analysis where we have compared the three algorithms on a test set containing 1440 projects with 50 activities and 5 renewable resources each. A detailed description of the remaining ProGen/max control parameters chosen can be found in De Reyck and Herroelen (1998*b*). We have imposed a limit t_{cpu} of 3 and 30 seconds on the maximum running time of the branch-and-bound algorithms, which refers to a Pentium personal computer operating at 200 MHz (for comparison purposes, the computation times have been scaled according to the clock pulse ratio by a factor of 0.3 for De Reyck and Herroelen's branch-and-bound algorithm and by a factor of 2.5 for the priority-rule method). Since De Reyck and Herroelen (1998*b*) only report on the number of instances for which the

branch-and-bound algorithm has completed the enumeration within the respective time limit, the values p_{opt} and p_{ins} and the values p_{nopt} and p_{unk} have been aggregated.

Table 3.5. Performance of algorithms for the net present value problem with renewable resources

Algorithm	t_{cpu}	p_{opt}	p_{ins}	p_{nopt}	p_{unk}
De Reyck and Herroelen (1998*b*)	3 s	58.1 %		41.9 %	
	30 s	75.5 %		24.5 %	
Neumann and Zimmermann (2002)	3 s	79.1 %	4.4 %	16.5 %	0.0 %
	30 s	85.1 %	4.4 %	10.5 %	0.0 %
Selle and Zimmermann (2003)	3 ms	1.0 %	4.4 %	94.6 %	0.0 %

Not surprisingly, the branch-and-bound algorithm by Neumann and Zimmermann (2002) seems to be more efficient than the earlier algorithm by De Reyck and Herroelen (1998*b*). The improvement upon the latter algorithm is probably to be attributed to the tremendous difference in the time needed for solving the relaxations. The dual method typically runs in a small fraction of the time that is required for re-optimizing from scratch the minimizer with the primal steepest descent method after the addition of a minimal delaying mode to the current relation. Moreover, the primal method is by far less time-consuming than the recursive-search procedure (see Table 3.3). The priority-rule method provides feasible schedules within a very short amount of time. The small proportion p_{opt} of instances, however, for which the optimal objective function value computed by the branch-and-bound algorithm of Neumann and Zimmermann (2002) can be found, indicates that the low computational effort is paid for by some loss of quality. Nevertheless, experience with the project duration problem documented in Franck et al. (2001*b*) suggests that priority-rule methods may constitute a valuable alternative to exact procedures when coping with projects comprising hundreds of activities. Finally, we notice that we do not give a deviation Δ_{lb} from some lower bound lb on the minimum objective function value because the latter quantity may be positive, zero, or negative. The development of a suitable index measuring the mean remaining error of suboptimal solutions for this type of problem seems to be an open issue in literature.

Starting from the representation of minimizers of a convex objective function on relation polytopes as spanning forests G of the project network N, Schwindt (2000*b*) has developed a neighborhood function for local search procedures (see also Neumann et al. 2003*a*). Similarly to the steepest descent algorithm from Subsection 3.2.2, the arcs of forest G correspond to active temporal or precedence constraints. G is decoded into the corresponding time-feasible schedule by computing a local minimizer S on the relation polytope $\mathcal{S}_T(\rho)$

where ρ is the relation in set V^a arising from the arcs of G that belong to precedence constraints (precedence arcs, for short). Two types of neighborhood operations are considered, which transform forest G into a neighboring forest G'. If S is feasible, G' results from G by deleting some precedence arc. Otherwise, a precedence arc may be deleted or a new precedence arc may be added for which both the initial and terminal nodes are contained in a forbidden active set for S. The reason why precedence arcs may also be cancelled even if S is not resource-feasible is that due to maximum time lags, it may be necessary to perform backtracking before attaining a feasible solution. When some precedence arc is deleted from G, the new minimizer of f is determined by applying the primal method starting at S. In case a precedence arc is added to G, the dual method is used.

We have tested a simple randomized best-fit search implementation (cf. Kolisch and Hartmann 1999) of this approach for the **total earliness-tardiness cost problem with renewable resources**. At each iteration the algorithm moves to the best neighboring forest. The quality of a forest G is evaluated according to the objective function value $f(S)$ of the corresponding schedule S and its degree of infeasibility measured in terms of the excessive workload $\sum_{k\in\mathcal{R}^\rho}\int_0^d (r_k(S,t)-R_k)^+dt$. In order to avoid cycling, the quality is randomly biased. Each time the local search gets stuck in a deadlock where S is not yet resource-feasible and no additional precedence arc can be added to G without generating a cycle of positive length in the corresponding relation network $N(\rho)$, we return to the best schedule found thus far. 10 % of the computation time is allotted to the branch-and-bound algorithm by Schwindt (2000*c*) for the computation of an initial feasible schedule serving as starting-point for the local search. If the branch-and-bound procedure fails in finding a feasible solution within the imposed time limit, the search starts at the minimizer of f on set $\mathcal{S}_T$.

The results for the branch-and-bound method and the best-fit search procedure are given in Table 3.6. They have been obtained for the test set with 90 instances comprising 100 activities and 5 renewable resources already used for the analysis of the algorithms for the time-constrained problem (see Table 3.4). Again, the tests have been performed on a 200 MHz Pentium personal computer.

Table 3.6. Performance of algorithms for the earliness-tardiness problem with renewable resources

Algorithm	t_{cpu}	p_{opt}	p_{ins}	p_{nopt}	p_{unk}	Δ_{lb}
Schwindt (2000*c*)	3 s	3.3 %	13.3 %	67.8 %	15.6 %	6.6 %
	30 s	5.6 %	13.3 %	70.0 %	11.1 %	6.5 %
	100 s	5.6 %	13.3 %	71.1 %	10.0 %	6.4 %
Schwindt (2000*b*)	83.4 s	3.3 %	13.3 %	75.6 %	7.8 %	6.0 %

Comparing the results from Tables 3.1 and 3.6 suggests that the earliness-tardiness problem is much more difficult to solve to optimality than the project duration problem. The mean deviation Δ_{lb} from the lower bound lb_0 arising from the resource relaxation, however, indicates that the quality of the schedules found is comparable to those computed for the project duration problem. This deviation can be further decreased by stopping the enumeration after a short amount of time and subsequently executing the best-fit search procedure based on the neighborhood function of Schwindt (2000*b*).

We conclude the subsection by considering the **capital-rationed net present value problem**, where the project is executed with a limited budget. In that case, the funds available for disbursement depend on the initial budget (possibly plus a credit line) and the difference of all past progress payments and paying outs. This situation frequently occurs in the building industry, where the receipts from completed subprojects serve to finance succeeding subprojects. It is readily seen that the cash balance can be interpreted as a cumulative resource with infinite storage capacity $\overline{R}$ and a safety stock of $\underline{R} = 0$. The initial inventory r_0 equals the project budget, and the resource requirements r_i of events $i \in V^e$, $i \neq 0$ coincide with the cash flows c_i^f. This project scheduling problem has been treated in an early paper by Doersch and Patterson (1977), who have devised an integer programming formulation based on time-indexed binary variables x_{it} being equal to one if $t = S_i$ and zero, otherwise. The objective function then reads $\sum_{i \in V} \sum_{t=ES_i}^{LS_i} c_i^f e^{-\alpha t} x_{it}$, and the resource constraints can be written as

$$\sum_{i \in V} \sum_{t'=ES_i}^{\min(t, LS_i)} c_i^f x_{it'} \geq \underline{R} \quad (t = 0, 1, \ldots, \overline{d})$$

A priority-rule method for solving the problem has been proposed by Smith-Daniels et al. (1996). The priority values are based on delay penalties, which arise from solving the dual of the time-constrained problem where the objective function is replaced by its first-order Taylor expansion (as it has been shown by Russell 1970, the dual then represents a transshipment problem).

Schwindt (2000*a*) has addressed the capital-rationed problem as a net present value problem with cumulative-resource constraints. His branch-and-bound algorithm is based on the enumeration scheme from Subsection 3.2.1, and the relaxations at the enumeration nodes are solved by the dual flattest ascent method discussed in Subsection 3.2.3. Kimms (2001*a*), Sect. 8.2, has proposed a mixed-integer linear program for a generalization of the problem setting where residual cash is lent from one period to the next and several projects from a given portfolio are considered simultaneously. The objective is to select the projects to be performed from the portfolio and to schedule the selected projects in a way that the cash balance at planning horizon $\overline{d}$ is maximized. Kolisch (1997) has investigated a variant of this problem where in addition, cash can be borrowed at an interest rate of $\alpha' \geq \alpha$ but only

one project is considered. For a critique of the underlying assumptions of this model we refer to Kimms (2001*a*), Sect. 8.1.

Table 3.7 shows the results of an experimental performance analysis comparing the branch-and-bound algorithm with the CPLEX 6.0 MIP solver processing Doersch and Patterson's integer programming formulation. The four test sets used consist of 90 instances each with 10, 20, 50, or 100 activities. For the projects with 10 or 20 activities, the `emphasis` parameter of the MIP solver has been put to `optimality`, whereas for the projects with 50 and 100 activities, this parameter has been chosen to be `feasibility`. The MIP solver and the branch-and-bound algorithm have been stopped after a maximum computation time of 100 seconds on a Pentium personal computer with 200 MHz clock pulse.

Table 3.7. Performance of algorithms for the net present value problem with one cumulative resource

Algorithm	n	p_{opt}	p_{ins}	p_{uopt}	p_{unk}
Doersch and Patterson (1977)	10	73.3 %	13.3 %	0.0 %	13.3 %
	20	50.0 %	0.0 %	7.8 %	42.2 %
	50	0.0 %	0.0 %	5.6 %	94.4 %
	100	0.0 %	0.0 %	0.0 %	100.0 %
Schwindt (2000*a*)	10	74.4 %	25.6 %	0.0 %	0.0 %
	20	74.4 %	25.6 %	0.0 %	0.0 %
	50	75.6 %	16.7 %	1.1 %	6.7 %
	100	65.6 %	8.9 %	5.6 %	20.0 %

The analysis clearly demonstrates the suitability of the cumulative-resource concept for solving this type of problems. Whereas the MIP solver is only capable of solving small problem instances of academic interest, the branch-and-bound algorithm terminates the enumeration within 100 seconds for almost 75 % of the projects with 100 activities. The instances with 10 and 20 activities are all either solved to optimality or shown to be unsolvable. It is worth noting that in contrast to the case of renewable resources (see Table 3.5), the difficulty resides rather in finding a feasible schedule than in proving optimality. Thus, developing advanced search strategies to overcome this difficulty may constitute a valuable field of future research.