Chapter 2

SHORTEST PATH PROBLEMS WITH RESOURCE CONSTRAINTS

Stefan Irnich
Guy Desaulniers

Abstract In most vehicle routing and crew scheduling applications solved by column generation, the subproblem corresponds to a shortest path problem with resource constraints (SPPRC) or one of its variants.

This chapter proposes a classification and a generic formulation for the SPPRCs, briefly discusses complex modeling issues involving resources, and presents the most commonly used SPPRC solution methods. First and foremost, it provides a comprehensive survey on the subject.

1. Introduction

For more than two decades, column generation (also known as branch-and-price when embedded in a branch-and-bound framework) has been successful at solving a wide variety of vehicle routing and crew scheduling problems (see e.g. Desrosiers et al., 1995; Barnhart et al., 1998; Desaulniers et al., 1998), and most chapters in this book). In most of these applications, the master problem of the column generation method is a (possibly generalized) set partitioning or set covering problem with side constraints, where most of the variables, if not all, are associated with vehicle routes or crew schedules. These route and schedule variables are generated by one or several subproblems, each of them corresponding to a *shortest path problem with resource constraints* (SPPRC) or one of its variants. The SPPRC has contributed to the success of the column generation method for this class of problems for three main reasons. Firstly, through its resource constraints, it constitutes a flexible tool for modeling complex cost structures for an individual route or schedule, as well as a wide variety of rules that define the feasibility of a route or a

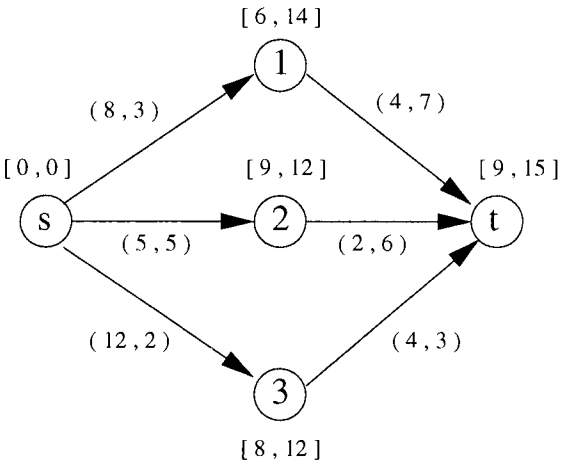

Figure 2.1. A small SPPRC example

schedule. Secondly, because it does not possess the integrality property (i.e., there may be a positive gap between its optimal value and that of its linear relaxation) as discussed in Desrosiers et al. (1984), the column generation approach can derive tighter bounds than those obtained from the linear relaxation of arc-based formulations. Thirdly, there exist efficient algorithms at least for some important variants of the SPPRC.

The SPPRC was introduced in the Ph.D dissertation of Desrochers (1986) as a subproblem of a bus driver scheduling problem. It consists of finding a shortest path among all paths that start from a source node, end at a sink node, and satisfy a set of constraints defined over a set of resources. A resource corresponds to a quantity, such as the time, the load picked-up by a vehicle, or the duration of a break in a work shift, that varies along a path according to functions, called resource extension functions (REFs). A REF is defined for every arc in the network and every resource considered. It provides a lower bound on the value that the corresponding resource can take at the head node of the corresponding arc, given the values taken by all the resources at its tail node. The resource constraints are given as intervals, called resource windows, which restrict the values that can be taken by the resources at every node along a path. Such a constraint is defined for every node in the network and every resource considered.

Figure 2.1 provides an SPPRC example that involves the resource *time*. The source and sink nodes are denoted by s and t, respectively. Each arc (i, j) bears a two-dimensional vector: The first component t_{ij}

provides the travel time (duration) of using the arc, while the second c_{ij} indicates the cost associated with it. Given a value T_i taken by the resource at a node i (T_i is said to be the visiting time at node i), the REF for an arc (i,j) is defined as $f_{ij}(T_i) = T_i + t_{ij}$, i.e., it computes the (*earliest*) *arrival time* at node j when starting at node i at time T_i. The resource window $[a_i, b_i]$ associated with each node i is specified in brackets beside it. It indicates at what time node i can be visited. If the arrival time of a path ending at a node i exceeds b_i, then this path is deemed infeasible. Otherwise, it is feasible even if its arrival time precedes a_i since waiting at a node is allowed, that is, the visiting time at node i can be greater than the arrival time at this node.

In the example of Figure 2.1, three paths link the source node s to the sink node t. The first path $P_1 = (s, 1, t)$, denoted by the sequence of nodes visited, is resource-feasible since it is possible to find visiting times along that path which satisfy all resource constraints. Indeed, setting $T_s = 0$ (the only feasible value at node s), it is easy to see that the arrival times ($T_1 = 8$ and $T_t = 12$) at nodes 1 and t provided by the appropriate REFs ($f_{s1}(T_s)$ and $f_{1t}(T_1)$) are all feasible with respect to the resource windows. The second path $P_2 = (s, 2, t)$ is also resource-feasible. However, waiting is needed at node 2 since the arrival time provided by $f_{s2}(0) = 5$ is smaller than $a_2 = 9$. In this case, the visiting time T_2 can be set at 9, and the subsequent visiting time T_t at 11, respectively. Finally, the third path $P_3 = (s, 3, t)$ is not resource-feasible since, along that path, $T_s = 0$, $T_3 \geq f_{s3}(0) = 12$, and the earliest arrival time at node t is $f_{3t}(12) = 16$. Hence, the resource window $[9, 15]$ at node t cannot be met. Since the cost of P_1 (3+7 = 10) is smaller than the cost of P_2 (5 + 6 = 11), the former path is optimal with respect to cost. However, path P_2 has a smaller earliest arrival time at node t. If the network in Figure 2.1 were only a sub-network within a bigger network, then extending path P_2 to a node could be feasible but extending P_1 could be infeasible.

This gives us a first glance at the core of SPPRC's difficulty. The SPPRC is very close to a multi-criteria problem. In the following we will consider both criteria, time and cost, as resources. Paths are *uncomparable* when one path is better than a second path in one criterion and worse in another criterion. Resource constraints make it necessary to consider all uncomparable paths that arrive at a node, since resource constraints might forbid extending any subset of these paths but allow an extension of the others.

The two-resource SPPRC, better known as the shortest path problem with time windows (SPPTW), was first studied in Desrosiers et al. (1983, 1984). The resource *cost* is unconstrained while the resource *time*

is restricted by corresponding time windows. Desrochers (1986) generalized the SPPTW to the case with several resources. Since then, several variants of the SPPRC have appeared in the literature. For instance, Ioachim et al. (1998) proposed the SPPTW with time dependent linear costs at the nodes and Dumas et al. (1991) the SPPTW with pickups and deliveries.

The contribution of this chapter is three-fold. Firstly, it presents a classification of the SPPRC variants and provides a generic SPPRC formulation that includes all variants studied so far (Section 2). Secondly, it discusses non-trivial modeling issues for the SPPRC (Section 3). Finally, it surveys the most important papers on this subject, namely, those introducing a new variant of the SPPRC (Section 2) or proposing an interesting methodological contribution (Section 4).

2. Classification of the SPPRCs

The intention of this section is to provide a generic formulation for a comprehensive class of shortest path problems with resource constraints presented in the literature so far. Variants of the SPPRC, which we consider, are extensions of the classical shortest path problem, where the cost is replaced by multi-dimensional resource vectors, which are accumulated along paths and constrained at intermediate nodes. Different types of SPPRCs can be classified by

(i) the way in which resources are accumulated, leading to different definitions of resource feasible paths,

(ii) the existence of additional path-structural constraints excluding specific paths, e.g., non-elementary paths,

(iii) the objective,

(iv) and the underlying network.

We state all SPPRCs on a digraph $G = (V, A)$, where V and A are non-empty sets of nodes and arcs, respectively. A *path* $P = (e_1, \ldots, e_p)$ is a finite sequence of arcs (some arcs may occur more than once) where the head node of $e_i \in A$ is identical to the tail node of $e_{i+1} \in A$ for all $i = 1, \ldots, p-1$. For the sake of convenience, we assume that G is simple so that a path can be written as $P = (v_0, v_1, \ldots, v_p)$ with the understanding that $(v_{i-1}, v_i) \in A$ holds for all $i \in \{1, \ldots, p\}$. The *length* of this path is p.

2.1 Resource feasible paths

The description of *feasible paths* provides a basis for the generic definition of the SPPRC. In the following, we distinguish between feasibility *w.r.t. resources* and feasibility *w.r.t. path-structural constraints*. This section focuses on the first aspect while path-structural constraints are discussed in the next section.

Resource constraints can be formulated by means of (*minimal*) *resource consumptions* and *resource intervals* (e.g., the travel times t_{ij} and time windows $[a_i, b_i]$ in the SPPTW). Let R be the number of resources. A vector $T = (T^1, \ldots, T^R)^\top \in \mathbb{R}^R$ is called a *resource vector* and its components *resource variables* (remark: $x^\top$ denotes the transposed vector to the vector x). T is said to be not greater than (i.e., *dominates*) $S = (S^1, \ldots, S^R)^\top \in \mathbb{R}^R$ if the inequality $T^i \leq S^i$ holds for all components $i = 1, \ldots, R$. We denote this by $T \leq S$. For two resource vectors a and b the interval $[a, b]$ is defined as the set $\{T \in \mathbb{R}^R \colon a \leq T \leq b\}$.

Resource intervals, also called *resource windows*, associated with a node $i \in V$ are denoted by $[a_i, b_i]$ with $a_i, b_i \in \mathbb{R}^R$, $a_i \leq b_i$. The changes in the resource consumptions associated with an arc $(i, j) \in A$ are given by a vector $f_{ij} = (f_{ij}^r)_{r=1}^R$ of so-called *resource extension functions* (REFs). A REF $f_{ij}^r \colon \mathbb{R}^R \to \mathbb{R}$ depends on a resource vector $T_i \in \mathbb{R}^R$, which corresponds to the resource consumption accumulated along a path from s to i, i.e., up to the tail node i of arc (i, j). Hence, the result $f_{ij}(T_i) \in \mathbb{R}^R$ can be interpreted as a resource consumption accumulated along the path $(s, \ldots, i, j)$. "Classical" SPPRCs, like the SPPTW presented in the introduction, only consider REFs of the form

$$f_{ij}^r(T_i) = T_i^r + t_{ij}^r \tag{2.1}$$

where t_{ij}^r are constants associated with the arc (i, j). Classical REFs are separable by resources, i.e., there exist no interdependencies between different resources. The more general definition of REFs provides a powerful instrument for modeling practically relevant resource interdependencies.

Instead of giving an implicit MIP-formulation for the SPPRC, we state the resource constraints by considering individual paths. The reason for this is that node repetitions within a path (which are allowed in our path definition) prohibit to model resource consumptions by individual resource variables associated with a node. For a given path $P = (v_0, v_1, \ldots, v_p)$, one has to refer to the $p + 1$ different *positions* $i = 0, 1, \ldots, p$. A path P is *resource-feasible* if there exist resource vectors $T_i \in [a_{v_i}, b_{v_i}]$ for all positions $i = 0, 1, \ldots, p$ such that $f_{v_i, v_{i+1}}(T_i) \leq T_{i+1}$ holds for all $i = 0, \ldots, p - 1$. $\mathcal{T}(P)$ is defined as the set of all feasible

resource vectors at the last node v_p of $P = (v_0, v_1, \ldots, v_p)$, i.e.,

$$\mathcal{T}(P) = \{T_p \in [a_{v_p}, b_{v_p}] \colon \exists T_i \in [a_{v_i}, b_{v_i}], f_{v_i,v_{i+1}}(T_i) \leq T_{i+1} \text{ for all } i = 0, \ldots, p-1\}. \quad (2.2)$$

Let $\mathcal{F}(u, v)$ be the set of all resource-feasible paths from a node u to a node v. Note that $P \in \mathcal{F}(u, v)$ holds if and only if $\mathcal{T}(P) \neq \varnothing$.

2.2 Path-structural constraints

Path-structural constraints can model further requirements concerning the feasibility of paths, which are not covered by resources. Such additional requirements might either be an integral part of a feasible path's definition or be implied by branching rules, which come up in the context of branch-and-price and require modifications of the pricing problem. Sometimes, these modifications cannot be handled by simply removing some arcs or nodes of the underlying network. In order to specify those constraints, we need some definitions. An *elementary path* is a path in which all nodes are pairwise different. Contrarily, a *cycle* is a path $(v_0, v_1, \ldots, v_p)$ of length $p > 1$ having $v_0 = v_p$. We call any cycle of length less than or equal to k a *k-cycle.*

The following SPPRC variants have been proposed in the literature and defined according to path-structural constraints. Let $\mathcal{G}$ be the set of all paths feasible with respect to these constraints.

For the *elementary SPPRC* (ESPPRC), $\mathcal{G} = \{\text{elementary paths}\}$. On acyclic graphs, all paths are elementary so that SPPRC and ESPPRC coincide. In general (i.e., for networks with cycles), the ESPPRC has been identified to be $\mathcal{NP}$-hard in the strong sense (Dror, 1994) and has been first studied and solved by Beasley and Christofides (1989). In many vehicle routing applications the pricing problem is an ESPPRC. Feillet et al. (2004); Chabrier (2002); Rousseau et al. (2003) solved ESPPRC pricing problems in the context of the vehicle routing problem with time windows (VRPTW). These approaches are known for their very tight lower bounds computed by the LP-relaxation of the VRPTW set-partitioning master program.

For the SPPRC, $\mathcal{G} = \{\text{all paths}\}$, that is, no path-structural constraints are imposed. The SPPRC occurs as a subproblem in numerous vehicle and crew scheduling problems which are most of the time formulated over acyclic time-space networks (see Desrosiers et al., 1984; Vance et al., 1997; Desaulniers et al., 1998; Gamache et al., 1999)).

Since the ESPPRC is very hard to solve (in some cases it is prohibitively hard), classical solution approaches for vehicle routing problems which are formulated over *cyclic graphs* are also based on the corre-

sponding non-elementary SPPRC, because it can be solved using pseudo-polynomial algorithms (see Section 4.1). Influential contributions which rely on this idea were Desrosiers et al. (1986); Desrochers et al. (1992); Desrosiers et al. (1995). However, while solving the enclosing problem by branch-and-price, this subproblem relaxation sometimes leads to weak lower bounds and possibly impractical large branch-and-bound trees.

For the *SPPRC with k-cycle elimination* (SPPRC-k-cyc), $\mathcal{G} = \{k\text{-cycle-free paths}\}$. A compromise between solving the ESPPRC and the SPPRC is to forbid cycles of small length. Several examples of VRPTW instances, e.g., taken from the benchmark library of Solomon (1987), show that cycle elimination for small values of k can substantially improve the master program lower bounds. This justifies an additional effort to eliminate cycles (compared to solving a pure SPPRC) while the corresponding ESPPRC is practically impossible to solve. The case $k = 2$ was first analyzed by Houck et al. (1980) and used in the VRPTW context by Kolen et al. (1987); Desrochers et al. (1992). Irnich and Villeneuve (2003) recently proposed an algorithm for the general case of $k \geq 2$.

For the *SPPRC with forbidden paths* (SPPRCFP), $\mathcal{G} = \{\text{all paths}\} \setminus \mathcal{G}_{\text{forbidden}}$ where $\mathcal{G}_{\text{forbidden}}$ is a set of forbidden paths. This set is implicitly defined as the set of all paths that contain at least one element of a finite set of pre-specified sub-paths. Villeneuve and Desaulniers (2000) introduced this type of SPPRC which occurs two-fold in the context of branch-and-price. First, in some applications one wants to branch so that a route or schedule is excluded from the (restricted) master program (see Desaulniers et al., 2002b; Arunapuram et al., 2003). This makes it necessary to also exclude the corresponding path from being generated by the SPPRC pricing procedure. Second, some constraints might be impossible or very hard to model with resources. Instead of considering them directly, one iteratively solves relaxed SPPRCs to get tentative solutions, which are excluded from the SPPRC by means of forbidden paths as long as not all constraints are respected. Examples of hard-to-model constraints stem from aircrew scheduling applications, see e.g. Fahle et al. (2002).

Two additional types of constraints, *precedence constraints* and *pairing constraints*, are important in the pickup and delivery context. Given two nodes $i, j \in V$, a path P fulfills the (i, j)-pairing constraint if node i occurs as often as node j in P (possibly P contains none of them). A path P fulfills the (i, j)-precedence constraint if P contains no subpath connecting j with i. The *SPPRC with pickups and deliveries* (SPPRCPD) is a subproblem of the vehicle routing problem with time windows, pickups and deliveries (see Dumas et al., 1991; Desaulniers et al.,

2002a). In this problem, transportation requests $i \in I$ must be satisfied where a request requires a pickup at an origin i^+ and a delivery at a destination i^-. Consequently, the SPPRCPD contains an (i^+, i^-)-pairing and an (i^+, i^-)-precedence constraint for each request $i \in I$.

In a branch-and-price context, each node and each arc represent a (possibly empty) sequence of tasks, where a *task* (e.g., a flight leg, a train segment, or a crew pairing) is associated with a set partitioning constraint in the master problem. A task can be part of several sequences and can therefore be represented by several nodes and arcs. For any path $P = (v_0, v_1, \ldots, v_p)$ there is a (uniquely defined) *task sequence* $W(P)$ given by the concatenation of the sequences of tasks of v_0, (v_0, v_1), v_1, $(v_1, v_2), \ldots, (v_{p-1}, v_p)$, v_p. All of the above path-structural constraints might also be formulated w.r.t. the task sequences. For instance, the task-ESPPRC considers only paths P for which $W(P)$ does not contain task repetitions or the task-SPPRC-2-cyc does not allow paths having a 2-cycle in $W(P)$.

Several branching rules proposed in the literature impose additional constraints on how two given tasks have to be covered by the paths. The branching rules of Ryan and Foster (1981) decide whether two tasks i and j are covered by the same path or by different paths. Hence, one branch is simply an (i, j)-pairing constraint. The other branch is an (i, j)-*anti-pairing constraint* which forbids tasks i and j to be together in $W(P)$, i.e., $\mathcal{G} = \{P\colon i \notin W(P) \text{ or } j \notin W(P)\}$. Similarly, the inter-task constraints (introduced in Desrochers and Soumis (1989)) decide whether two given tasks i and j are performed consecutively or not. In this case, an (i, j)-*follower constraint* guarantees on one branch that, for each path $P \in \mathcal{G}$, $W(P)$ contains task i followed by task j or none of these tasks. On the other branch, an (i, j)-*non-follower constraint* only allows paths $P \in \mathcal{G}$ for which $W(P)$ does not contain task i followed by task j.

Summing up the definitions of resource feasibility and path-structural constraints, we know that the set $\mathcal{F} = \bigcup_{v \in V} (\mathcal{F}(s, v) \cap \mathcal{G})$ contains all feasible paths to a one-to-all SPPRC problem.

2.3 Objectives and generic SPPRC formulation

The objective of the SPPRC is formulated by means of a resource vector at the last nodes of feasible paths. Recall that in general, for a single path $P \in \mathcal{F}$ there exist many feasible choices for the resource vectors $T \in \mathcal{T}(P)$. Problems whose objective depends only on a single resource, called *cost* resource, are normally one-to-one shortest path problems with a source node s and a sink node t. They can be formulated

as follows:

$$\min_{P \in \mathcal{F}(s,t) \cap \mathcal{G}} \left(\min_{T \in \mathcal{T}(P)} T^{\text{cost}} \right). \tag{2.3}$$

Computing the minimum cost of a path $P = (v_0, \ldots, v_p)$ requires the determination of feasible resource vectors $T_0, \ldots, T_p$ along the path. Similarly to the feasibility problem $\mathcal{T}(P) \neq \varnothing$ discussed above, this can be a hard problem. In contexts with time windows, Dumas et al. (1990) optimized the cost of a given path for time-dependent convex inconvenience costs at all nodes.

A much more general formulation of the SPPRC is based on considering the set of Pareto-optimal resource vectors. For a given set $M \subset \mathbb{R}^R$, an element $m \in M$ is *Pareto-optimal* if $x \not\leq m$ holds for all $x \in M, x \neq m$. It means that none of the cones $x^{\llcorner}$ for $x \in M, x \neq m$ contain a Pareto-optimal point m, where a *cone* $T^{\llcorner}$ is defined as $\{S \in \mathbb{R}^R \colon S \geq T\}$. For $v \in V$, let $PO(v)$ be the set of Pareto-optimal vectors in $\bigcup_{P \in \mathcal{F}(s,v) \cap \mathcal{G}} \mathcal{T}(P)$. The SPPRC can be formulated as follows.

> **Generic SPPRC:** Find for each node $v \in V$ and for each Pareto-optimal resource vector $T \in PO(v)$ one feasible (representative) s-v-path $P \in \mathcal{F}(s,v) \cap \mathcal{G}$ having $T \in \mathcal{T}(P)$.

For the sake of convenience, we call the representative path P a *Pareto-optimal path.* Since all solutions to a problem $\min_{m \in M} \alpha^\top \cdot m$ for a non-negative weight vector $\alpha \in \mathbb{R}^R_+$, $\alpha \neq 0$ are Pareto-optimal points of M, the generic SPPRC formulation also solves all problems of the form

$$\min_{P \in \mathcal{F}(s,t) \cap \mathcal{G}} \left(\min_{T \in \mathcal{T}(P)} \alpha^\top T \right) \tag{2.4}$$

for any weight vector $\alpha \in \mathbb{R}^R_+$. Problem (2.3) is a special case of (2.4).

2.4 Properties of $\mathcal{T}(P)$

We will now study properties of the set $\mathcal{T}(P)$ for a fixed path $P = (v_0, v_1, \ldots, v_p)$ under different assumptions concerning the REFs. Knowing $\mathcal{T}(P)$ and its structure is essential to (efficiently) resolve the following two basic tasks:

- Given a path P. Is P resource feasible, i.e., $P \in \mathcal{F}(v_0, v_p)$ or not?
- Given the prefix $P' = (v_0, \ldots, v_{p-1})$ of $P = (v_0, \ldots, v_{p-1}, v_p)$, compute $\mathcal{T}(P)$ using $\mathcal{T}(P')$.

Furthermore, compact implicit representations of $\mathcal{T}(P)$ are substantial for checking if a path P (or any of its extensions) is or might be a Pareto-optimal path. For instance, efficient dominance checks in the context of

dynamic programming are based on representing $\mathcal{T}(P)$ by either using a single Pareto-optimal point $T(P)$ or a function $g_P(\cdot)$ to describe the set of Pareto-optimal points in $\mathcal{T}(P)$, see Section 4.1.

Before discussing different cases, we state the following universal property: If $T \in \mathcal{T}(P)$ then $T^{\llcorner} \cap [a_{v_p}, b_{v_p}] \subseteq \mathcal{T}(P)$, i.e., the set $\mathcal{T}(P)$ contains the cone, restricted to the resource interval, generated by each point in this set.

Classical SPPRC and non-decreasing REFs. In the classical SPPRC the set $\mathcal{T}(P)$ has a simple representation as a cone restricted by $[a_{v_p}, b_{v_p}]$. Let $P_i = (v_0, \ldots, v_i)$, $i = 0, \ldots, p$ be the prefix of P of length i. Each set $\mathcal{T}(P_i)$ has a unique cone-defining element $T(P_i) \in \mathcal{T}(P_i)$ such that $\mathcal{T}(P_i) = T(P_i)^{\llcorner} \cap [a_{v_i}, b_{v_i}]$ holds. The resource vector $T(P_i)$ can be recursively computed by

$$\begin{aligned} &T(P_0) = a_{v_0} \quad \text{and} \\ &T(P_i) = \max\{a_{v_i}, f_{v_{i-1},v_i}(T(P_{i-1}))\} \quad \text{for all } i \in \{1, \ldots, p\}. \end{aligned} \tag{2.5}$$

The same is true when all REFs are non-decreasing functions, meaning that each $f_{ij}^r(T_i^1, T_i^2, \ldots, T_i^R)$ is a non-decreasing function in one variable T_i^k, when the other $R-1$ components are kept fixed. Under these assumptions $\mathcal{T}(P)$ is still a cone. Formula (2.5) computes $T(P)$ with $T(P)^{\llcorner} \cap [a_{v_p}, b_{v_p}] = \mathcal{T}(P)$ efficiently.

As a consequence, the generic SPPRC formulation can be simplified as follows.

> **Generic SPPRC with non-decreasing REFs:** Find for each node $v \in V$ one feasible representative s-v-path $P \in \mathcal{F}(s,v) \cap \mathcal{G}$ for which $T(P)$ is Pareto-optimal in $\{T(Q) \colon Q \in \mathcal{F}(s,v) \cap \mathcal{G}\}$.

Formulation (2.4) can then be re-written as $\min_{P \in \mathcal{F}(s,t) \cap \mathcal{G}} \alpha^{\top} T(P)$.

Linear REFs. If the REFs are linear but not necessarily non-decreasing, it is easy to see that $\mathcal{T}(P)$ is a bounded polyhedron. The description of the polyhedron $\mathcal{T}(P)$ (e.g., by its extreme points) can get more and more complicated the longer the path P is (see Ioachim et al., 1998) and Section 4.1.2).

For instance, consider the path $P = (1, 2)$, $R = 2$ resources, resource intervals $[a_1, b_1] = [0, 1]^2$ and $[a_2, b_2] = [0, 1] \times [-1, 1]$ and the REF $f_{12}(T_1^1, T_1^2) = (T_1^1, T_1^2 - T_1^1)$. It is easy to see that $\mathcal{T}(P)$ is $\{(T_2^1, T_2^2) \in [0, 1] \times [-1, 1] \colon T_2^2 \geq -T_2^1\}$. There exists no element $T \in \mathcal{T}(P)$ such that $\mathcal{T}(P) \subseteq T^{\llcorner}$ holds. Note that all vectors $T = (\lambda, -\lambda)$ for $\lambda \in [0, 1]$ are Pareto-optimal points of $\mathcal{T}(P)$.

General REFs. For arbitrary REFs, checking whether $P \in \mathcal{F}(u,v)$ or equivalently $\mathcal{T}(P) \neq \varnothing$ holds or not can be an $\mathcal{NP}$-hard problem. A known $\mathcal{NP}$-complete problem is the binary knapsack lower bound feasibility problem (KLBFP) (see Nemhauser and Wolsey, 1988): *Does there exist a feasible solution with profit at least lb for a given lower bound lb to the knapsack problem* $\max \sum_{i=1}^{n} p_i x_i$, $\sum_{i=1}^{n} w_i x_i \leq C$, $x \in \{0,1\}^n$? One can easily transform this decision problem into an SPPRC with three resources: Negative profit, weight, and decision. Let $G = (V, A)$ be a line graph with nodes $V = \{0, 1, \ldots, n\}$ and arcs $A = \{(0,1), (1,2), (2,3), \ldots, (n-1, n)\}$. Let $[a_0, b_0] = [0,0] \times [0, C] \times [0,1]$, $[a_n, b_n] = [-\infty, -lb] \times [0, C] \times [0,1]$, and $[a_i, b_i] = [-\infty, 0] \times [0, C] \times [0,1]$ be the resource windows at all nodes $i \in V \setminus \{0, n\}$. Define the REFs to be $f_{i-1,i}(p, w, x) = (p, w, 0)$ for $x = 0$, and $f_{i-1,i}(p, w, x) = (p - p_i, w + w_i, 0)$ for $x \neq 0$. The answer to the KLBFP is "yes" if and only if $\mathcal{T}(P) \neq \varnothing$ for the path $P = (0, 1, \ldots, n)$.

2.5 Underlying network

The SPPRCs can also be differentiated according to whether or not their underlying network is acyclic or cyclic. The existence of cycles implies that there exist infinitely many different paths in G (not necessarily feasible w.r.t. resource and path-structural constraints). Thus, the SPPRC might be unbounded. In the following, we exclude these cases from our consideration.

The following discretization of $G = (V, A)$ formally makes the underlying network acyclic. If there exists at least one non-decreasing resource r (i.e., $f_{ij}^r(T_i) - T_i^r > 0$, or $t_{ij}^r > 0$ in the classical SPPRC with $f_{ij}^r(T_i) = T_i^r + t_{ij}^r$ for all $(i,j) \in A$, e.g., the resource time in many applications) it is possible to transform (V, A) into an acyclic time-space network. Each node $v \in V$ is replaced by several copies $\text{copy}^1(v), \ldots, \text{copy}^p(v)$ corresponding to a time discretization of the resource interval for r. Nevertheless, this transformation is only a formal device, e.g., used in the unified model of Desaulniers et al. (1998). Cycles of the original network correspond with paths visiting two or more copies of the same original node. Solving the ESPPRC in G is, therefore, equivalent to solving an SPPRC with task-cycle elimination in the discretized network.

3. Modeling issues

The modeling of standard constraints like capacity constraints, path length restrictions and time windows is obvious from the introduction. Other simple examples can be found in Vance et al. (1997); Gamache et al. (1999); Desaulniers et al. (1999). This section will, therefore, focus

Table 2.1. Resource intervals and REFs for task-related constraints.

Constraint	Type	Resource interval $[a_i^r, b_i^r]$ for all $i \in V$	REF $f_{ij}^r(T_i)$ for all $(i,j) \in A$
(k,ℓ)-pairing	$R^=$	$[0,0]$ for $i = s,t$ $[-M,M]$ for $i \in V \setminus \{s,t\}$	$T_i^r + \delta_{ik} - \delta_{i\ell}$
(k,ℓ)-anti-pairing	$R^=$	$[0,0]$ for $i = s$ $[0,M]$ for $i = k$, $[-M,0]$ for $i = \ell$ $[-M,M]$ for $i \in V \setminus \{s,k,\ell\}$	$T_i^r + \delta_{ik} - \delta_{i\ell}$
(k,ℓ)-precedence	$R^\leq$	$[0, 1 - \delta_{ik}]$	$T_i^r + \delta_{i\ell}$
(k,ℓ)-pairing and precedence	$R^=$	$[0,0]$ for $i = s,k,t$ $[-1,-1]$ for $i = \ell$ $[-1,1]$ for all $i \in V \setminus \{s,t,k,\ell\}$	$T_i^r + \delta_{i\ell} - \delta_{ik}$
(k,ℓ)-follower and (k,ℓ)-non-follower	$R^=$	$[l(W(s)), l(W(s))]$ for $i = s$ $[0,N]$ for $i \in V \setminus \{s\}$	(see equation (2.6))

on non-trivial modeling issues, provide examples and give references to some relevant literature.

In some applications, one wants to model *exact resource consumptions* instead of *minimal resource consumptions*. For the SPPTW it means that waiting is not allowed so that the arrival time at each node is always identical to the visiting time. In general, the inequalities in (2.2) defining a resource-feasible path $P = (v_0, v_1, \ldots, v_p)$ have to be replaced by $T_{i+1}^r = f_{v_i,v_{i+1}}^r(T_i)$.By $R^=$ (resp. $R^\leq$) we denote the resources which force an equality (resp. inequality) in (2.2). However, as suggested in Gamache et al. (1998), a resource $r \in R^=$ might equivalently be replaced by two resources $r_1, r_2 \in R^\leq$ where the resource intervals and REFs for r_1 are identical to those for r while those for r_2 are $[a_i^{r_2}, b_i^{r_2}] = [-b_i^r, -a_i^r]$ and $\tilde{f}_{ij}^{r_2}(\tilde{T}_i) = -f_{ij}^r(\tilde{T}_i^1, \ldots, \tilde{T}_i^{r-1}, -\tilde{T}_i^{r_2}, \tilde{T}_i^{r+1}, \ldots, \tilde{T}_i^R)$ (the $\tilde{}$ symbol refers to the case with the r_1 and r_2 resources).

Section 2.2 has provided several examples of path-structural constraints. Most of them can be modeled with additional resources (one for each constraint) in a standard SPPRC. For the ESPPRC, Beasley and Christofides (1989) proposed to add to $R^\leq$ an additional resource r_v for each node $v \in V$. (For a compact notation, we use the Kronecker-symbol with $\delta_{ij} = 1$ if $i = j$, and $\delta_{ij} = 0$, otherwise.) The resource intervals are defined as $[a_i^{r_v}, b_i^{r_v}] = [0, 1 - \delta_{si}]$ for all $i \in V$ and the REFs by $f_{ij}^{r_v}(T_i) = T_i^{r_v} + \delta_{iv}$ for all $(i,j) \in A$.

Table 2.1 gives an overview of how (anti-)pairing constraints, precedence constraints, and (non-)follower constraints can be modeled by

means of resources. In this table, M is a sufficiently large positive integer. For the first group (pairing, anti-pairing, and precedence) we assume that a single task is associated with each node. Note that the modeling proposed for the (k,ℓ)-pairing and precedence constraints is equivalent to the set component proposed by Dumas et al. (1991) for the SPPRCPD.

If a single task is associated with each node, follower and non-follower constraints simply imply the removal of some of the arcs (see e.g. Desrochers and Soumis (1989)). Therefore, we present these constraints for the case that sequences of tasks are associated with arcs and nodes. We assume that tasks are numbered from 1 to N, the last task of any non-empty task sequence $W(\cdot)$ is denoted by $l\big(W(\cdot)\big)$. For empty task sequences one defines $l\big(W(\varnothing)\big) = 0$.

All follower and non-follower constraints can be modeled with a single resource r, where $T_i^r \in \{1,\ldots,N\}$ means that the last task of the task sequence of the current path $(s,\ldots,v_i)$ was the one with number T_i^r. $T_i^r = 0$ means that the current path has an empty task sequence. The definition of the corresponding REFs is:

$$f_{ij}^r(T_i) = \begin{cases} T_i^r & \text{if } W\big((i,j),j\big) = \varnothing \\ l\big(W\big((i,j),j\big)\big) & \text{if } T_i^r \neq 0,\ W\big((i,j),j\big) \neq \varnothing, \\ & \quad \text{and } \big(T_i^r, W\big((i,j),j\big)\big) \text{ feasible} \\ l\big(W\big((i,j),j\big)\big) & \text{if } T_i^r = 0,\ W\big((i,j),j\big) \neq \varnothing, \\ & \quad \text{and } W\big((i,j),j\big) \text{ feasible} \\ -1 & \text{otherwise.} \end{cases} \tag{2.6}$$

The strength of the non-classical REF concept is that it allows multiple resources to depend on each other. In several applications such as the aircrew pairing problem Vance et al. (1997), the cost of a path depends on several resources. A second example of non-trivial dependent REFs stems from the capacity constraints of the *VRPTW with simultaneous pickups and deliveries*, see Min (1989); Desaulniers et al. (1998). Here, each customer $i \in V \setminus \{s,t\}$ has demanded for delivery q_i^d and for pickup q_i^p. A vehicle of capacity Q starts at the depot s with the entire delivery demand of the tour loaded. It services each customer (pickup after delivery) so that the vehicle reaches the final depot t having the entire pickup demand on board. A feasible path (route) is one in which the pickups of already visited nodes plus the deliveries of the following customers do not exceed the vehicle capacity on any arc traveled. The feasibility problem is modeled with two dependent resources $r_p, r_{\max} \in R^{\leq}$, where the resource variable $T_i^{r_p}$ is *demand already picked* (directly after node i) and $T_i^{r_{\max}}$ is the *maximum load* in the vehicle on the path from s

to i. Obviously, one has $[a_i^{r_p}, b_i^{r_p}] = [a_i^{r\max}, b_i^{r\max}] = [0, Q]$ for all $i \in V$ and $f_{ij}^{r_p}(T_i^{r_p}, T_i^{r\max}) = T_i^{r_p} + q_j^p$ for all $(i,j) \in A$. For the maximum load, one has non-linear but non-decreasing REFs $f_{ij}^{r\max}(T_i^{r_p}, T_i^{r\max}) = \max\{T_i^{r_p} + q_j^p, T_i^{r\max} + q_j^d\}$. It means that the maximum load at node j (following node i) is either the entire pickup demand at the end of the path, computed by $T_i^{r_p} + q_j^p$, or results from the maximum load on the sub-path $(0, \ldots, i)$ to which the delivery of j has to be added.

The modeling of other non-linear resource consumptions is straightforward, e.g., *soft time windows* (see Dumas et al., 1990), *load-dependent travel costs* or *time-dependent travel times* (connections (i,j) with different travel durations depending on the time of the day). Complex schedule regulations and their modeling can be found in Desaulniers et al. (1997); Vance et al. (1997).

Another non-trivial example of dependent resources is the computation of the *minimal waiting time* for an SPPTW path. With the notation for the SPPTW given in the introduction, the total waiting time along path $P = (v_0, v_1, \ldots, v_p)$ is given by $T_p - T_0 - \sum_{i=1}^{p} t_{i-1,i}$. Desaulniers and Villeneuve (2000) showed that three resources with non-decreasing REFs are enough to compute both the earliest arrival time and the minimal waiting time (or equivalently, an associated waiting cost).

4. Solution methods

This section describes different methodologies developed for solving the SPPRCs, namely, dynamic programming which has been used extensively, Lagrangean relaxation, constraint programming, and heuristics. It also presents a graph modification approach for the SPPRCFP.

4.1 Dynamic programming and labeling algorithms

Dynamic programming solution approaches for the SPPRC systematically build new paths, starting from the trivial path $P = (s)$, by extending paths one-by-one into all feasible directions. Their efficiency depends on the ability to identify and discard paths which are not useful either to build a Pareto-optimal set of paths or to be extended into Pareto-optimal paths. Discarding non-useful paths is achieved by a dominance sub-algorithm based on dominance rules, which strongly depend on the path-structural constraints and the properties of the REFs.

For the sake of efficiency, paths in the dynamic programming algorithms are encoded by *labels*. Paths sharing a common prefix are represented by using a single chain of labels for their common prefix. This

is implemented with the help of a tree data structure in which a label corresponding to path $P = (v_0, \ldots, v_{p-1}, v_p)$ is directly linked back to the label of the prefix path $(v_0, \ldots, v_{p-1})$ (see e.g. Ahuja et al., 1993, for an introduction to labeling algorithms). Beside encoding the path itself, the label typically stores a representation of $\mathcal{T}(P)$, e.g., given by the unique resource vector $T(P)$ in case of non-decreasing REFs. In Ioachim et al. (1998) a more complex representation of $\mathcal{T}(P)$ is stored in the labels, while Irnich and Villeneuve (2003) store additional (compressed) information to accelerate the dominance algorithm.

In order to formalize the above ideas, we need some definitions. For a given path $P = (v_0, v_1, \ldots, v_p)$ we call $v(P) = v_p$ the *resident node* of P. A path $P = (v_0, v_1, \ldots, v_p)$ is a *feasible extension* of path $Q = (w_0, w_1, \ldots, w_q)$ if $(Q, P) = (w_0, \ldots, w_q, v_0, \ldots, v_p) \in \mathcal{F}(w_0, v_p) \cap \mathcal{G}$. The set of all feasible extensions is $\mathcal{E}(Q) = \{P\colon (Q, P) \in \mathcal{F}(w_0, v(P)) \cap \mathcal{G}\}$.

Labeling algorithms rely on the manipulation of two sets. The first set $\mathcal{U}$ is the set of *unprocessed paths*, which have not yet been extended. The second set $\mathcal{P}$ is the set of *useful paths*. Useful paths $P \in \mathcal{P}$ have already been processed. They have been identified to be Pareto-optimal or might be prefixes of Pareto-optimal paths (note that Pareto-optimal paths might have prefixes which are not Pareto-optimal, see Section 4.1.2). Both sets, $\mathcal{U}$ and $\mathcal{P}$, change dynamically in the course of the labeling algorithm.

One can identify two basic procedures invoked by the labeling algorithm (see the pseudo-code below). In the *path extension step* an unprocessed path $Q \in \mathcal{U}$ is chosen, all feasible extensions (Q, v) with $v \in V$ are constructed and added to $\mathcal{U}$, while Q itself is removed from $\mathcal{U}$. Thus, the extension step replaces one element of $\mathcal{U}$ by all of its feasible one-node extensions. Once processed, an element is transferred to the set $\mathcal{P}$. If possible, the *dominance algorithm* reduces the sets $\mathcal{U}$ and $\mathcal{P}$. Its goal is to accelerate the overall labeling procedure by limiting the number of necessary extension steps.

The path extension step and the dominance algorithm maintain the following invariant: *The useful paths $\mathcal{P}$ and all extensions of unprocessed paths $\mathcal{U}$ together contain a solution of the SPPRC.* Recall from Section 2.3 that an SPPRC solution is not necessarily unique since it contains representatives taken from a set of desired solutions, e.g., one path for each Pareto-optimal resource vector. Therefore, let Σ be the set of all different solutions of an SPPRC, where each element $\mathcal{S} \in \Sigma$ is a set of paths, e.g., Pareto-optimal paths. The above invariant is

$$\exists \mathcal{S} \in \Sigma\colon \mathcal{S} \subseteq \{(Q, P)\colon Q \in \mathcal{U}, P \in \mathcal{E}(Q)\} \cup \mathcal{P}. \tag{2.7}$$

The algorithm is initialized with $\mathcal{U} = \{P_0\}$ and $\mathcal{P} = \varnothing$ where $P_0 = (s)$ is the trivial path. Each path $P = (v_0, v_1, \ldots, v_p) \in \mathcal{F}$ results from an extension of P_0, i.e., $(v_1, \ldots, v_p) \in \mathcal{E}(P_0)$. Hence, condition (2.7) holds for the initialization. Obviously, the path extension step also maintains the invariant. The crucial point is to define dominance rules in such a way that the dominance algorithm also respects (2.7). We focus on that aspect in Section 4.1.2. By doing so, the algorithm finally terminates with an $\mathcal{S} \subseteq \mathcal{P}$ for some $\mathcal{S} \in \Sigma$. In a post-processing *filtering step* Pareto-optimal solutions can be extracted from $\mathcal{P}$.

Generic Dynamic Programming SPPRC Algorithm {
(* Initialize *)
SET $\mathcal{U} = \{(s)\}$ and $\mathcal{P} = \varnothing$
WHILE $\mathcal{U} \neq \varnothing$ DO
(* Path extension step *)
CHOOSE a path $Q \in \mathcal{U}$ and REMOVE Q from $\mathcal{U}$
FORALL arcs $(v(Q), w) \in A$ of the forward star of $v(Q)$ DO
IF $(Q, w) \in \mathcal{F}(s, w) \cap \mathcal{G}$ THEN ADD (Q, w) to $\mathcal{U}$
ADD Q to $\mathcal{P}$
(* Dominance step *)
IF (* any condition *)
APPLY dominance algorithm to paths from $\mathcal{U} \cup \mathcal{P}$ ending at some node v
(* Filtering step *)
FILTER $\mathcal{P}$, i.e., identify a solution $\mathcal{S} \subseteq \mathcal{P}$
}

Several remarks should be made.

1 If one performs path extension steps only, but no dominance steps, the result is $\mathcal{P} = \mathcal{F}$, i.e., the algorithm computes all feasible paths.

2 The path extension step leaves the freedom to choose paths $Q \in \mathcal{U}$ according to different processing strategies. These path selection strategies can lead to *label setting* or *label correcting* algorithms depending on the underlying network and the REFs. These issues will be discussed in Section 4.1.1.

3 The dominance algorithm can be applied at any time in the course of the algorithm. In order to keep the effort small, it makes sense to delay the dominance algorithm to a point when there is a chance to remove several of the paths at the same time, before they are processed in the path extension step.

The dominance rules strongly depend on the problem at hand. Section 4.1.2 discusses the impact of different path-structural constraints and classical, non-decreasing, special or general REFs.

4 There exist efficient algorithms for the filtering step to identify, e.g., Pareto-optimal paths (see Bentley, 1980; Kung et al., 1975).

4.1.1 Label setting and label correcting algorithms. The defining property of a *label setting algorithm* is that those labels chosen to be extended (in the path extension step) are kept until the end of the labeling process. They will not be identified as discardable in subsequent calls of the dominance algorithm. Labeling algorithms that do not guarantee this behavior are called *label correcting algorithms.* The general ideas of label setting as well as label correcting algorithms in the context of the one-dimensional shortest path problem (SPP) are, for instance, explained in the book of Ahuja et al. (1993).

An acyclic network $G = (V, A)$ naturally gives rise to label setting algorithms if paths are treated (that is, chosen and extended) according to a topological order of their resident nodes. More precisely, the above generic algorithm loops over the topologically sorted nodes $v = s, v_2, \ldots, v_{|V|}$, applies the dominance algorithm to the paths $\{P \in \mathcal{U} \cup \mathcal{P} : v(P) = v\}$ resident at the current node v, and extends those paths who survive the dominance process into all feasible directions.

It is possible to mimic an acyclic network for the treatment of labels if the resource consumptions for at least one resource r are strictly positive, i.e., $f_{ij}^r(T_i) - T_i^r > 0$ holds for all $(i, j) \in A$ and all $T_i \in [a_i, b_i]$. In this case, the labeling algorithm chooses unprocessed paths $Q \in \mathcal{U}$ with minimum (or "small") $T(Q)^r$ for extension first. It is guaranteed that paths Q already treated only produce extensions (Q, P) with $T(Q, P)^r > T(Q)^r$. Hence, newly generated paths cannot enforce the elimination of already treated paths. Desrochers and Soumis (1988) used the concept of generalized buckets to identify paths with small value $T(Q)^r$.

Label correcting algorithms solve shortest path instances with negative arc lengths. The existence of negative resource consumptions $f_{ij}^r(T_i) - T_i^r$ for an arc (i, j) and all resources r (i.e., negative t_{ij}^r for the classical SPPRC) means that the strategy of treating paths in a strictly increasing ord er of their resource vectors has to be replaced by a more flexible processing strategy. The well-known Ford-Bellman label correcting algorithm for the SPP adds newly generated labels to the end of a queue and extends labels one-by-one starting with the label currently at the top of the queue. Powell and Chen (1998) have presented a more sophisticated generalized label correcting strategy for the SPPRC, which is directly applicable to the general SPPRC case.

4.1.2 Dominance rules and dominance algorithms. Efficient dominance rules have been described for the SPPRC, ESPPRC

and SPPRC-k-cyc with non-decreasing REFs. Recall that in these cases each path $P \in \mathcal{F}(s,v)$ has a unique resource vector $T(P) \in \mathcal{T}(P)$, which is the only Pareto-optimal point of $\mathcal{T}(P)$.

Dominance rules identify paths Q to be non-useful in the following sense: Q is neither necessary to describe the set of Pareto-optimal solutions $PO\big(v(Q)\big)$, nor feasible extensions $Q' \in \mathcal{E}(Q)$ lead to paths (Q,Q') necessary to construct $PO\big(v(Q')\big)$. Such a path Q can be discarded. Typically, dominance rules identify non-useful paths by comparing $T(Q)$ and $\mathcal{E}(Q)$ with the corresponding values $T(P)$ and $\mathcal{E}(P)$ of paths P resident at node $v(P) = v(Q)$. We discuss the cases SPPRC, ESPPRC, SPPRC-2-cyc, and SPPRC-k-cyc with non-decreasing REFs in detail.

SPPRC. Given two different paths $P, Q \in \mathcal{U} \cup \mathcal{P}, v(P) = v(Q)$ with $T(P) \leq T(Q)$, the dominance algorithm can discard path Q while keeping P, which results from the following two arguments. First, $T(P) \leq T(Q)$ means that $T(Q)$ is not necessary to represent Pareto-optimal paths ending at $v(Q)$. Second, one has to investigate possible extensions of Q. The fact $T(P) \leq T(Q)$, the absence of any path-structural constraints and the non-decreasing REF imply $\mathcal{E}(P) \supseteq \mathcal{E}(Q)$. Therefore, any $Q' \in \mathcal{E}(Q)$ fulfills $(P,Q') \in \mathcal{F}$ and $T(P,Q') \leq T(Q,Q')$. There do not result any Pareto-optimal resource consumptions from extensions of Q which could not have been built using extensions of P. Hence Q can be discarded.

Note that dominance rules are sensitive to the occurrence of paths with identical resource vectors. Therefore, one has to distinguish between *dominance* and *discarding dominated paths*. Two paths $P, Q \in \mathcal{F}(s,v)$ with $T(P) = T(Q)$ dominate each other but only one of these two can be eliminated (while the other one is kept). (Irnich and Villeneuve, 2003) propose techniques to resolve ambiguity and analyze them for the SPPRC and SPPRC-k-cyc cases.

ESPPRC. In presence of path-structural constraints, the relation $T(P) \leq T(Q)$ does not necessarily imply the relation $\mathcal{E}(P) \supseteq \mathcal{E}(Q)$. For the ESPPRC, the reason is that paths $P \in \mathcal{G}$ can only be extended to nodes not already visited. We denote the set of visited nodes by $V(P)$. A restricted dominance rule for the ESPPRC allows to discard path Q if $T(P) \leq T(Q)$ and $V(P) \subseteq V(Q)$ since both conditions together imply $\mathcal{E}(P) \supseteq \mathcal{E}(Q)$. Beasley and Christofides (1989) modeled the sets $V(P)$ for paths $P \in \mathcal{F}$ by one additional resource for each node of V.

Feillet et al. (2004) improved the idea of Beasley and Christofides. They interpreted the set $V(P)$ differently as the *"set of nodes which cannot be visited any more"*. By analyzing the resource vector $T(P)$

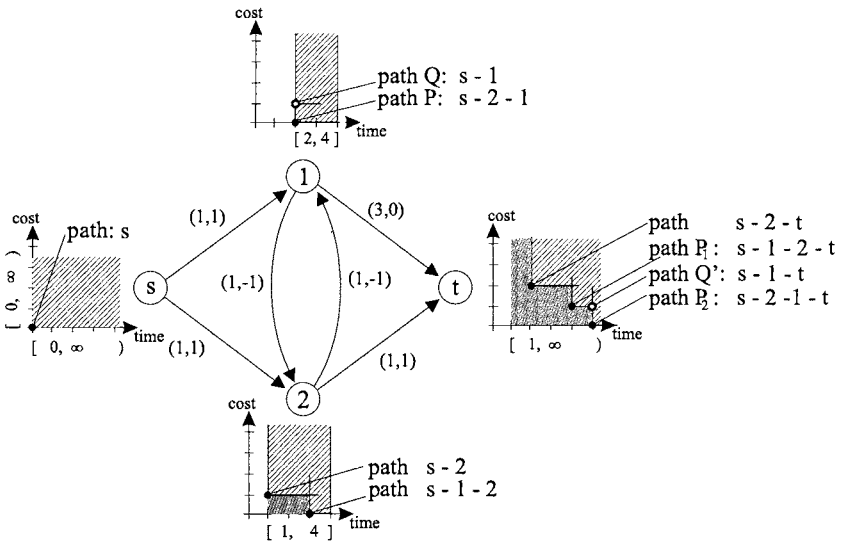

Figure 2.2. Example of an SPPTW with 2-cycle elimination.

they identified additional unvisited nodes which are impossible to reach (e.g., because of current time, time window constraints and non-negative travel times). These nodes are added to the set $V(P)$ to form the set $\overline{V}(P)$. As a result, the above dominance rule based on the "extended" sets $\overline{V}(P)$ can eliminate more paths.

SPPRC-2-cyc. An informal description of a dominance rule for the 2-cycle elimination case is the following: *Keep only a Pareto-best path P_1 and a second-best path P_2 which is extended from a different predecessor node.* For any path $P = (v_0, \ldots, v_{p-1}, v_p)$ with $p \geq 1$, the node v_{p-1} is called the *predecessor node* and denoted $\text{pred}(P)$. It is easy to see that the SPPRC dominance rule applies to paths P, Q, $v(P) = v(Q)$, $T(P) \leq T(Q)$ having identical predecessor nodes. Kohl (1995); Larsen (1999) showed that if $\mathcal{E}(P)$ does not contain the one-node path $\big(\text{pred}(P)\big)$, i.e., the dominating path P cannot be extended to its predecessor node, the SPPRC dominance rule also remains valid. Contrarily, given three different paths P_1, P_2, Q, $v(P_1) = v(P_2) = v(Q)$, $T(P_1), T(P_2) \leq T(Q)$ with different predecessors $\text{pred}(P_1) \neq \text{pred}(P_2)$, one can discard path Q while keeping P_1 and P_2. The proof of this rule is based on the fact that $\text{pred}(P_1) \neq \text{pred}(P_2)$ implies $\mathcal{E}(P_1) \cup \mathcal{E}(P_2) \supseteq \mathcal{E}(Q)$.

An example of an SPPTW with 2-cycle elimination is shown in Figure 2.2 and illustrates the two above-mentioned dominance rules.

First, at node 1 the paths P and Q fulfill $T(P) \leq T(Q)$. Since $\text{pred}(P) \neq \text{pred}(Q)$ it is not allowed to eliminate Q. This is substantial because the dominated path $Q = (s, 1)$ is a prefix of the Pareto-optimal path $P_1 = (s, 1, 2, t)$ at the sink t. The path-structural constraints imply that some dominated paths, like Q, are still useful paths. Second, path Q' at node t can be discarded because the two dominating paths P_1 and P_2 have different predecessor nodes (alternatively, because P_2 and Q' have the same predecessor node).

SPPRC-k-cyc. Handling the k-cycle elimination case for $k \geq 3$ needs sophisticated data structures (see Irnich and Villeneuve, 2003). In essence, the dominance rule efficiently checks whether

$$\mathcal{E}(Q) \subseteq \bigcup_{P \in \mathcal{P} \cup \mathcal{U}:\ T(P) \leq T(Q), v(P) = v(Q)} \mathcal{E}(P) \tag{2.8}$$

holds, i.e., all extensions of dominating paths cover the extensions of Q. A path Q for which (2.8) holds can be discarded. There exists a finite representation of the right hand side of (2.8), which uses up to $(k-1)!^2$ vectors (so-called *set forms*) with $\binom{k}{2}$ entries. Moreover, these set forms can be used to efficiently encode and update the relation (2.8) so that the evaluation of (2.8) can be performed in constant time. From a complexity point of view, the main result of this dominance rule is that the maximum number of paths stored in $\mathcal{P} \cup \mathcal{U}$ grows by a factor $\alpha(k)$ compared to the classical SPPRC. The factor $\alpha(k)$ is independent of the size of the underlying network and bounded by $\alpha(k) \leq k(k-1)!^2$.

SPPTWTC. Another case where efficient dominance rules have been described is the shortest path problem with time windows and time costs (SPPTWTC)(see Ioachim et al., 1998). An SPPTWTC instance is uniquely defined by the SPPTW data, i.e., travel costs c_{ij}, travel times t_{ij}, and time windows $[a_j, b_j]$, together with arbitrary node costs $w_j \in \mathbb{R}$ (positive as well as negative) for the nodes $j \in V$. Visiting the node j at time T_j^{time} causes additional *time costs or profits* of $w_j T_j^{\text{time}}$. Hence, depending on the sign of w_j it is advantageous to visit node j as early or as late as possible. When negative and positive time costs occur together at the nodes of a path, the determination of feasible visiting times T_j^{time} with minimum overall cost is an optimization problem in itself.

Formally, the SPPTWTC is a two-resource problem with a resource *time* and a time-dependent resource *cost*. The REFs for time are given

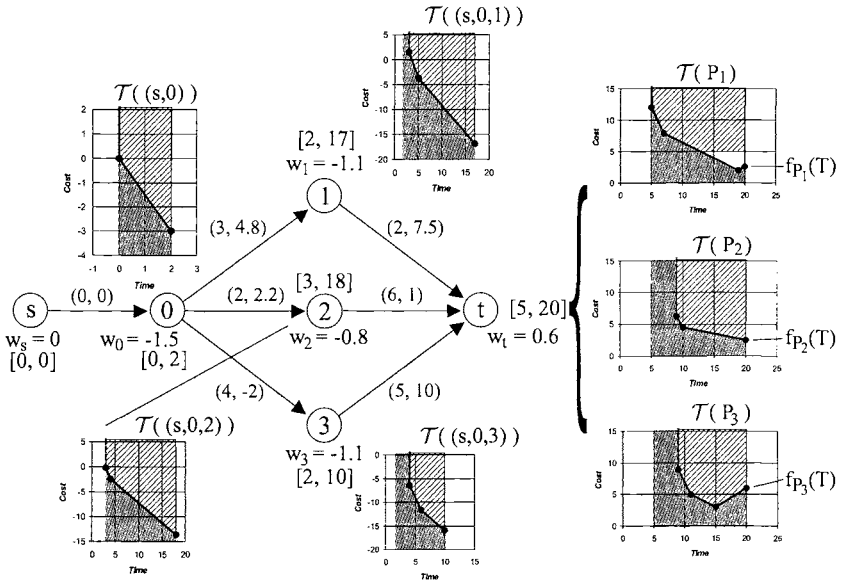

Figure 2.3. Example of an SPPTWTC: Travel time and travel cost are given as pairs (t_{ij}, c_{ij}) for each arc (i, j), time windows $[a_i, b_i]$ and linear node costs w_i are given for each node i, paths ending at node t are $P_1 = (s, 0, 1, t)$, $P_2 = (s, 0, 2, t)$, and $P_3 = (s, 0, 3, t)$.

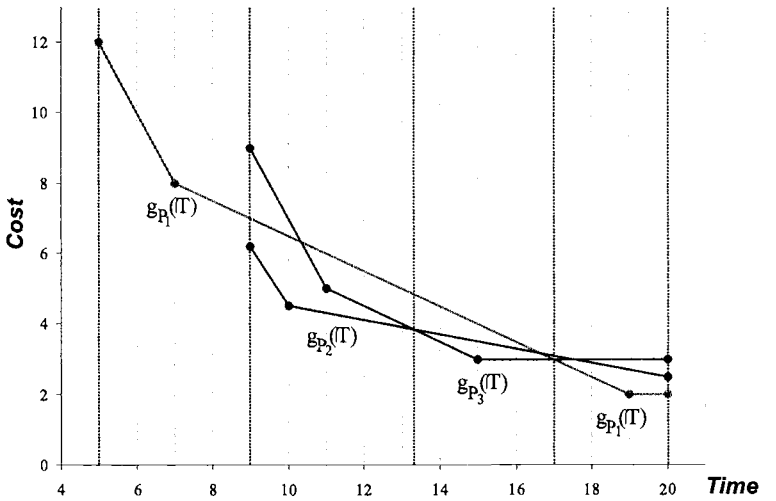

Figure 2.4. Piecewise linear cost functions representing $\mathcal{T}(P)$ for the paths $P_1 = (s, 1, t)$, $P_2 = (s, 2, t)$, and $P_3 = (s, 3, t)$.

by $f_{ij}^{\text{time}}(T_i^{\text{time}}) = T_i^{\text{time}} + t_{ij}$ and for cost by $f_{ij}^{\text{cost}}(T_j^{\text{time}}, T_i^{\text{cost}}) = T_i^{\text{cost}} + c_{ij} + w_j T_j^{\text{time}}$. This is a (minor) extension of the REF concept of Section 2 because f_{ij}^{cost} depends on a resource variable T_j^{time} of the node j (and not only on resource variables T_i at node i). Figure 2.3 shows a small SPPTWTC example. Next to each node, the resource space (a time-cost diagram) shows the set $\mathcal{T}(P)$ for each of the feasible paths P. Obviously, $\mathcal{T}(P)$ is a bounded polyhedron.

Dominance rules for the SPPTWTC were proposed by Ioachim et al. (1998). Although presented differently, their main ideas are the following. The set $\mathcal{T}(P)$ is determined by its lower envelope, which is a piecewise linear (cost) function $f_P(T^{\text{time}})$ with a maximum of $p+1$ pieces when P has length p (in the following we use $T = T^{\text{time}}$). The function $f_P(T)$ is convex and only its first strictly decreasing part is relevant for dominance (since the objective is to find the minimum-cost path, the nonnegative slope segments are useless (see Ioachim et al., 1998, p. 196). Hence, the relevant piecewise linear cost function is

$$g_P(T) = \begin{cases} f_P(T), & \text{for } T \leq \arg\min_T f_P(T) \\ f_*, & \text{for } T \geq \arg\min_T f_P(T) \end{cases}$$

with the minimum $f_* = \min_T f_P(T)$. Simple update formulas allow to compute $g_P(T)$ from $g_{P'}(T)$ when P' is the prefix path of $P = (P', v)$. A path Q can be discarded if there exists paths $P_1, \ldots, P_k$ ending at $v = v(Q) = v(P_1) = \cdots = v(P_k)$ with $\mathcal{T}(Q)^{\llcorner} \subseteq \bigcup_{i=1}^{k} \mathcal{T}(P_i)^{\llcorner}$ (for a set X the symbol $X^{\llcorner}$ denotes the set $\bigcup_{x \in X} x^{\llcorner}$). This dominance rule can be implemented by computing the minimum cost function $G_v(T) = \min_P g_P(T)$ over all paths P ending at node v. Each path Q with $v(Q) = v$ which does not contribute to the minimum cost function $G_v(T)$ can be discarded. Figure 2.4 shows the situation for the three paths P_1, P_2, and P_3 ending at node t from the above example. All paths P_1, P_2, P_3 contribute to $G_t(T)$, which is composed of four pieces imposed by $g_{P_1}(T)$ for $T \in [5, 9)$, $g_{P_2}(T)$ for $T \in [9, 13.\bar{3}]$, $g_{P_3}(T)$ for $T \in [13.\bar{3}, 17]$, and $g_{P_1}(T)$ for $T \in [17, 20]$. None of the paths are dominated by the other paths.

4.2 Lagrangean relaxation

The constrained shortest path problem (CSPP) is a specialized s-t-SPPRC with independent additive resource consumptions along arcs. The resource consumption is constrained only as a whole and not by individual resource intervals. The objective is to find a least-cost s-t-path with resource consumptions within a pre-specified interval. Among others, Beasley and Christofides (1989); Borndörfer et al. (2001) proposed to

solve the CSPP with Lagrangean relaxation for computing lower bounds and a tree search procedure exploiting these computed lower bounds. For the remainder of this section we assume that the underlying network $G = (V, A)$ is acyclic.

For a formal description of the CSPP, consider R different resources including *cost* as the last resource R with cost matrix $C = (c_{ij}) = (t^R_{ij})$. For the remaining resources, let $\mathcal{R} = (t^r_{ij}) \in \mathbb{R}^{(R-1)\times|A|}$ be the *resource consumption matrix* with non-negative consumptions t^r_{ij} for $r = 1, \ldots, R-1$. The REFs are $f^r_{ij}(T_i) = T_i + t^r_{ij}$ for all $r = 1, \ldots, R$ and $(i, j) \in A$, and the resource accumulation is $T_j = f_{ij}(T_i)$ whenever arc (i, j) is traversed. Lower bounds $l \in \mathbb{R}^{R-1}$ and upper bounds $u \in \mathbb{R}^{R-1}$ on the overall accumulated resource consumptions are implied by defining $[a_s, b_s] = [0, 0$, $[a_t, b_t] = [l, u]$, and $[a_i, b_i] = [0, u]$ at all other node $i \in V \setminus \{s, t\}$. For a given path P and its incidence vector $x \in \{0, 1\}^{|A|}$, the resource consumption is $\mathcal{R}x$ and the cost is $c^\top x$. P is feasible if $l \leq \mathcal{R}x \leq u$ holds. Borndörfer et al. (2001) have added a *goal value* $g \in [l, u]$ for the resource consumption $\mathcal{R}x$ to the formulation of Beasley and Christofides (1989). Slack and surplus variables z_+, z_- measure the deviation of $\mathcal{R}x$ from g, which is penalized by $p_+, p_- \in \mathbb{R}^{R-1}_{\geq 0}$. The CSPP can be stated as follows:

$$z_{\text{CSPP}} = \min c^\top x + p_-^\top z_- + p_+^\top z_+ \tag{2.9a}$$
$$\text{subject to } Ix = e_s - e_t \tag{2.9b}$$
$$\mathcal{R}x + z_+ - z_- = g \tag{2.9c}$$
$$(z_-, z_+) \leq (u - g, g - l) \tag{2.9d}$$
$$x \in \{0, 1\}^{|A|}, \quad z_-, z_+ \in \mathbb{R}^R_{\geq 0} \tag{2.9e}$$

Cost (2.9a) is a combination of accumulated travel costs and the penalty for the deviation of $\mathcal{R}x$ from g. Flow conservation constraints (2.9b) are given by means of the arc-node incidence matrix $I \in \{-1, 0, 1\}^{|V|\times|A|}$ and unit vectors $e_s, e_t \in \{0, 1\}^{|V|}$. They guarantee that $\{(i, j) \colon x_{ij} = 1\}$ forms a path in the acyclic network G. Constraints (2.9d) bounds the slack and surplus variables so that $l \leq \mathcal{R}x \leq u$ is ensured.

A Lagrangean relaxation of (2.9) can be obtained by relaxing the resource consumption constraints (2.9c). Let $\pi \in \mathbb{R}^{R-1}$ be an associated dual price vector. The Lagrangean dual of (2.9) is $\max_{\pi\in\mathbb{R}^{R-1}} z_{\text{DCSPP}}(\pi)$ where the Lagrangean subproblem decomposes into the following two parts:

$$z_{\text{DCSPP}}(\pi) = P(\pi) + B(\pi) + \pi^\top g \tag{2.10a}$$
$$\text{with } P(\pi) = \min(c^\top - \pi^\top \mathcal{R})x,$$

$$Ix = e_s - e_t, \quad x \in \{0,1\}^{|A|} \tag{2.10b}$$

$$\text{and} \quad B(\pi) = \min(p_+^\top - \pi^\top)z_+ + (p_-^\top + \pi^\top)z_-,$$
$$0 \leq z_- \leq u - g, \quad 0 \leq z_+ \leq g - l. \tag{2.10c}$$

The first part (2.10b) is an SPP, which can be solved with a label setting algorithm (see Ahuja et al., 1993). The second part (2.10c) is a minimization problem defined over a box, which is trivial to solve by inspection of the signs of the components of $(p_+^\top - \pi^\top)$ and $(p_-^\top + \pi^\top)$.

High quality solutions for the above Lagrangean dual formulation can be computed with any subgradient optimization method, e.g., a coordinate ascent method as in Borndörfer et al. (2001). The same authors proposed to use such a dual solution π^* and the dual solution of (2.10b) obtained for $\pi = \pi^*$ (i.e., a distance vector $(h_v(\pi^*))_{v \in V}$) to compute so-called *Lagrangean distance labels*:

$$g_v(\pi^*) = h_v(\pi^*) - h_s(\pi^*) + B(\pi^*) + g^\top \pi^*, \quad \text{for all } v \in V.$$

These labels are very useful to prune the search tree because of the following property. Let $x^1 \in \{0,1\}^{|A|}$ and $x^2 \in \{0,1\}^{|A|}$ be path incidence vectors of an s-v-path and a v-t-path, respectively. If $x = x^1 + x^2$ is a feasible CSPP path and $\pi^* \in \mathbb{R}^{R-1}$ a Lagrangean multiplier vector, then

$$z_{\text{CSPP}}(x) \geq g_v(\pi^*) + (c^\top - \pi^{*\top}\mathcal{R})x^2 \tag{2.11}$$

where $z_{\text{CSPP}}(x)$ denotes the cost of path x. The inequality means that if the right-hand side is non-negative then there exists no prefix path x^1 such that $x^1 + x^2$ has a negative (reduced) cost. Consequently, one should implement a tree search for finding negative (reduced) cost CSPP paths in G by systematically building v-t-paths x^2 starting at the sink node t. A tentative path x^2 can be discarded if the right-hand side of (2.11) becomes non-negative. Note that additional constraints that could not be considered in (2.9) can always be taken into account in the search phase.

4.3 Constraint programming

Constraint programming (CP) relies on a model which is defined by a set of variables, each with an initial domain, and a set of constraints. A CP approach is composed of a *search mechanism* to explore the solution space, a *domain reduction algorithm* for each constraint that tries to remove inconsistent values from the domains of the variables involved in that constraint, and a *propagation algorithm* that propagates these domain changes among the constraints. It allows to consider a wide

spectrum of constraints (algebraic and non-algebraic), including some that cannot be modeled using resources or simple path-structural constraints: For instance, an employee cannot work more than 8 hours in every 24-hour period. Within column generation approaches, CP has recently been used to tackle the SPPRC on an acyclic network (de Silva, 2001; Fahle et al., 2002) and the ESPPRC (Rousseau et al., 2003). In both cases, the goal is solely to find at least one feasible path with a negative (reduced) cost. This goal is modeled as a constraint (the cost of a feasible path must be negative), yielding a constraint satisfaction problem.

Fahle et al. (2002) considered an SPPRC on an acyclic network where a task, defined by a starting and an ending time, is associated with each node. They proposed a model where a boolean variable is associated with each node. Such a variable is set to *true* if the corresponding node is part of the path currently built. In this case, we will say that the node is *selected*. Additional variables are also used to specify, for instance, the minimal amount of rest to assign after each task. Their model includes simple constraints such as the boolean variables associated with two nodes whose tasks must be performed concurrently cannot be set at *true* simultaneously, or the total duration of the tasks associated with the selected nodes cannot exceed the maximum worked time in a schedule. Given a set of selected nodes, these two types of constraints can be used to fix some boolean variables to *false*.

In de Silva (2001), a different CP model is used. It involves variables to indicate the successor node next$[t]$ of each node t and variables to specify the amount of accumulated resource consumptions at each node. Nodes with next$[t] = t$ are not included in the current path. Path constraints model resource consumptions along the selected (partial) path, e.g., for the reduced cost, total working time, etc. Each time that a successor node is selected, the propagation algorithm is invoked, i.e., constraints are verified by solving an SPP for every unselected node of the underlying network. For instance, one can exclude a node (i.e., set next$[t] = t$) if the value of the path with the shortest worked time and passing through that node t and all selected nodes exceeds the maximum total worked time. A similar decision propagation based on the (reduced) cost of a path can also be executed. So-called *goals*, e.g., based on reduced cost shortest path computations, control how new tasks are added to the current partial path. The search tree is usually explored until a prespecified number of negative cost paths are found or until a time limit is reached.

For the ESPPRC, Rousseau et al. (2003) used a similar model with variables for the successor node and variables for the accumulated re-

source consumptions. Some of the constraints they consider are: All successor nodes must be different, no subtours are allowed, lower bounds provided by the resource REFs must be respected, the (reduced) cost of a feasible path must be negative. For verifying this last constraint, the authors compute a lower bound by solving an assignment problem. The choice of the next variable to branch on in the search tree is made in such a way to construct a path from the source to the sink node.

4.4 Heuristics

Even with sophisticated solution methods, solving an SPPRC instance might still be very time-consuming. In the column generation context, solving SPPRCs to proven optimality is only necessary to show that no negative reduced cost paths exist in the last pricing step. In preceding iterations it is sufficient to approximately solve the SPPRC, i.e., to compute any negative reduced cost feasible paths. That is the point where heuristics for the SPPRC come into play. In addition, they might be applied when the entire column generation problem is treated heuristically. In the following, we distinguish between three major areas of application for heuristics: *Pre-processing*, *dynamic programming*, and *direct search*.

Classical *pre-processing* techniques eliminate arcs and reduce the resource intervals (see e.g. Desrochers et al., 1992). The heuristic version of this idea is to solve a given SPPRC instance on a hierarchy of restricted networks, where each of the restricted networks contains only a limited number of arcs, e.g., defined by the $p > 0$ "nearest neighbors" of each node. Starting with the smallest p-nearest-neighbor network, one solves the associated SPPRC, and if no solution is found, one continues with the next p. This idea has been used in many implementations (e.g. Dumas et al. (1991); Savelsbergh and Sol (1998); Larsen (1999); Irnich and Villeneuve (2003)). Another idea is to replace some of the resources by less accurate resources to get an easier-to-solve SPPRC network. Gamache et al. (1999) gave the example where a restricted network measures time rounded up to the nearest hour while the exact global network uses minutes.

Dynamic programming heuristics are based on the techniques of Section 4.1 but heuristically accelerate the computation. For the VRPTW, Larsen (1999) used a so-called *forced early stop* rule to quit from the dynamic program when an adequate number of negative reduced cost columns has been found and a pre-defined number of labels has been generated. Chabrier (2002) tried to solve the ESPPRC by using the standard path extension step (i.e., not extending a path to a node already visited) with the stronger SPPRC dominance rule (i.e., only the

resource vectors are compared but not the visited nodes). Clearly, this procedure is quick but might fail to detect any negative reduced cost path. Therefore, he proposed to iteratively apply a dynamic programming procedure which combines the ESPPRC extension step with a gradually parametrizable dominance rule. A parameter *DomLevel* (between 0 and ∞) defines the length of a path after which the ESPPRC dominance rule is applied. If the partial path is shorter, the heuristic SPPRC dominance rule is applied. Larger values of *DomLevel* make the modified dynamic programming procedure substantially faster. The case *DomLevel* $= 0$ corresponds with the exact ESPPRC and is expected to be quite slow (especially for non-adjusted dual variables). Hence, starting with a large value for *DomLevel*, the dynamic programming algorithm with the modified dominance rule is iteratively applied with decreasing values of *DomLevel* until a negative reduced cost path is found (or the ESPPRC is solved exactly).

Finally, *direct search heuristics* are mainly based on local search. Such improvement procedures start from a given feasible path P and delete, insert, or replace nodes or exchange arcs in order to find an improving feasible path P' with smaller reduced cost. Note that after solving the restricted master program, the basic variables provide a set of paths with reduced cost 0 from which an improvement algorithm might start. Successful column generation applications which use these techniques can be found in Savelsbergh and Sol (1998); Xu et al. (2003).

4.5 A graph modification approach for the SPPRCFP

The graph modification approach for the SPPRCFP defined on a given network $G = (V, A)$ is not a solution method in itself but a method that manipulates G to obtain a new network $G' = (V', A')$ from which all forbidden paths are removed while the other paths of G are still feasible. One can then apply any of the proposed methods for the SPPRC to the network G' to solve the given SPPRCFP. Formally, let $\mathcal{H}$ be the set of forbidden sub-paths and let $\mathcal{G}_{\text{forbidden}} = \{(P, Q, P') \colon P, Q, P' \text{ paths}, Q \in \mathcal{H}\}$ so that $\mathcal{G} = \{ \text{ all paths}\} \setminus \mathcal{G}_{\text{forbidden}}$ is the set of all feasible paths for the SPPRCFP. The approach of Villeneuve and Desaulniers (2000) merges the original graph G with the state graph of a finite automaton, which identifies the infeasible subpaths in $\mathcal{H}$. We illustrate the procedure by an example in which G is given in Figure 2.5(a) and $\mathcal{H} = \{(1, 2, 4), (2, 1), (2, 3, 1)\}$.

The approach works in two stages. First, the algorithm of Aho and Corasick (1975) is used to construct the state graph $S = (V_S, A_S)$ of a

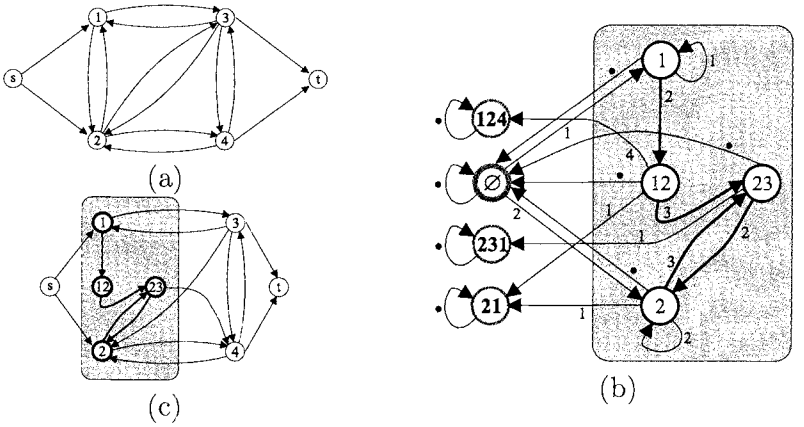

Figure 2.5. (a) Example network $G = (V, A)$ for the SPPRCFP. (b) State graph $S = (V_S, A_S)$ of a finite automaton which identifies the sub-paths of $\mathcal{H} = \{(1,2,4),(2,1),(2,3,1)\}$. "•" stands for any label $v \in V$ except those corresponding to the other out-arcs of the same node. (c) Resulting SPPRCFP network $G' = (V', A')$, node 12 corresponds with node 2 of the original network and node 23 with node 3.

finite automaton, which processes the nodes of a path P to detect the first sub-path in $\mathcal{H}$ it contains. The nodes (states) in V_S correspond to the prefixes of the sub-paths in $\mathcal{H}$, i.e., $V_S = \{\varnothing, 1, 12, 124, 2, 21, 23, 231\}$ in the example (see Figure 2.5(b)). Each time that a node of P is processed, the automaton performs a state transition. Possible transitions are represented by labeled arcs. There is an arc $(z_1, z_2) \in A_S$ labeled with $v \in V$ (v represents a possible node of P) that connects two different states z_1 and z_2 if $z_1 \notin \mathcal{H}$ and $z_2 = \big(\sigma(z_1), v\big)$, where $\sigma(z_1)$ is the longest (possibly empty) suffix of z_1 for which $\big(\sigma(z_1), v\big) \in V_S$. Further loops, i.e., $(124, 124)$, $(231, 231)$, and $(21, 21)$ guarantee that once a forbidden sequence has been detected, the automaton stays in the corresponding state. The remaining transitions connect a state z_1 back to the initial state $\varnothing$. The states $z \in \mathcal{H}$ indicate that a forbidden path has been detected.

Second, the original graph G has to be merged with the state graph S to produce a new graph $G' = (V', A')$. Prefixes $z \in V_S$ of length 1 are identified with the nodes of V. The new node set V' consists of all original nodes V and all nodes of V_S except the state $\varnothing$ and the states $z \in \mathcal{H}$. In the example, the new node set V' is $\{s, 1, 2, 3, 4, 12, 23, t\}$. In order to get the new arc set A', one has to join the sets A and $\{(z, z') \in A_S \colon z, z' \in V'\}$ and to perform three operations:

(i) remove all loops of the arc set A_S;

(ii) remove from the original arc set A the first arc of each sub-path in $\mathcal{H}$;

(iii) replace each transition $(z, \varnothing)$ of the finite automaton by a set of arcs (z, v) with $v \in V$ such that $(z, v) \notin V_S$ but $\big(\lambda(z), v\big) \in A$ where $\lambda(z)$ denotes the last node of the prefix z.

In the example, all loops and the arcs $(1,2)$, $(2,1)$ and $(2,3)$ are removed while the arc $(23,4)$ replaces the transition $(23,\varnothing)$. The new digraph $G' = (V', A')$ is depicted in Figure 2.5(c). A node z in V' represents the node $\lambda(z)$ in V so that paths in G' are in correspondence with paths in G. For instance, the path $(s, 1, 12, 23, 2, 4, t)$ corresponds with the feasible path $(s, 1, 2, 3, 2, 4, t)$ of the original network.

5. Conclusions

This survey has highlighted the richness of the SPPRC. In particular, it showed its great flexibility to incorporate a wide variety of constraints, yielding numerous SPPRC variants as well as diversified solution methods. We have given a new classification scheme and a generic formulation, which integrates the special purpose SPPRC formulations presented in the literature so far. Future research on the SPPRC will focus on developing more efficient exact and heuristic algorithms for some of the most difficult SPPRCs such as the ESPPRC or the SPPRC with general REFs. Additionally, with the application of column generation to a wider class of vehicle routing and crew scheduling problems, one should expect new variants of the SPPRC that will require the adaptation of existing solution methods or the development of new ones.

References

Aho, A. and Corasick, M. (1975). Efficient string matching: An aid to bibliographic search. *Journal of the ACM*, 18(6):333–340.

Ahuja, R., Magnanti, T., and Orlin, J.(1993). Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs, New Jersey.

Arunapuram, S., Mathur, K., and Solow, D.(2003). Vehicle routing and scheduling with full truck loads. *Transportation Science*, 37(2):170–182.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

Beasley, J. and Christofides, N.(1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394.

Bentley, J. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229.

Borndörfer, R., Grötschel, M., and Löbel, A. (2001). Scheduling duties by adaptive column generation. *Technischer Bericht* (ZIB-Report) 01-02, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Berlin.

Chabrier, A. (2002). Vehicle routing problem with elementary shortest path based column generation. *Technical Report*, ILOG, Madrid.

Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M. M., and Soumis, F. (1997). Crew pairing at Air France. *European Journal of Operational Research*, 97:245–259.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M., and Soumis, F. (2002a). VRP with pickup and delivery. In: *The Vehicle Routing Problem* (P. Toth and D. Vigo, D., eds.), Chapter 9, pp. 225–242. Siam, Philadelphia.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: *Fleet Management and Logistics* (T. Crainic and G. Laporte, eds.), Chapter 3, pp. 57–93. Kluwer Academic Publisher, Boston, Dordrecht, London.

Desaulniers, G., Desrosiers, J., Lasry, A., and Solomon, M. M. (1999). Crew pairing for a regional carrier. In: *Computer-Aided Transit Scheduling* (N. Wilson, ed.), Lecture Notes in Computer Science, Volume 471, pp. 19–41. Springer, Berlin.

Desaulniers, G., Langevin, A., Riopel, D., and Villeneuve, B. (2002b). Dispatching and conflict-free routing of automated guided vehicles: An exact approach. *Les Cahiers du GERAD* G-2002-31, HEC, Montréal, Canada. Forthcoming in: *International Journal of Flexible Manufacturing Systems.*

Desaulniers, G. and Villeneuve, D. (2000). The shortest path problem with time windows and linear waiting costs. *Transportation Science*, 34(3):312–319.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354.

Desrochers, M. and Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *Information Systems and Operations Research*, 26(3):191–212.

Desrochers, M. and Soumis, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation Science*, 23(1):1–13.

Desrochers, M. (1986). La Fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes. *Ph.D Thesis*, Centre de recherche sur les Transports, Publication #470, Université de Montréal, Canada.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In: *Handbooks in Operations Research and Management Science* (M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, eds.), Volume 8, *Network Routing*, Chapter 2, pp. 35–139. Elsevier, Amsterdam.

Desrosiers, J., Pelletier, P., and Soumis, F. (1983). Plus court chemin avec contraintes d'horaires. *RAIRO*, 17:357–377.

Desrosiers, J., Soumis, F., Desrochers, M., and Sauve, M. (1986). Methods for routing with time windows. *European Journal of Operational Research*, 23:236–245.

Desrosiers, J., Soumis, F., and Desrochers, M. (1984). Routing with time windows by column generation. *Networks*, 14:545–565.

de Silva, A. (2001). Combining constraint programming and linear programming on an example of bus driver scheduling. *Annals of Operations Research*, 108:277–291.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–978.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pick-up and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22.

Dumas, Y., Soumis, F., and Desrosiers, J. (1990). Optimizing the schedule for a fixed vehicle path with convex inconvenience costs. *Transportation Science*, 24(2):145–152.

Fahle, T., Junker, U., Karisch, S., Kohl, N., Sellmann, M., and Vaaben, B. (2002). Constraint programming based column generation for crew assignment. *Journal of Heuristics*, 8(1):59–81.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

Gamache, M., Soumis, F., and Marquis, G. (1999). A column generation approach for large-scale aircrew rostering problems. *Operations Research*, 47(2):247–263.

Gamache, M., Soumis, F., Villeneuve, D., Desrosiers, J., and Gélinas, E. (1998). The preferential bidding system at Air Canada. *Transportation Science*, 32(3):246–255.

Houck, D., Picard, J., Queyranne, M., and Vemuganti, R. (1980). The travelling salesman problem as a constrained shortest path problem: Theory and computational experience. *Opsearch*, 17:93–109.

Ioachim, I., Gélinas, S., Desrosiers, J., and Soumis, F. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31:193–204.

Irnich, S. and Villeneuve, D. (2003). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *Les Cahiers du GERAD* G-2003-55, HEC, Montréal, Canada.

Kohl, N. (1995). Exact methods for time constrained routing and related scheduling problems. *Ph.D Thesis*, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Kolen, A., Rinnooy-Kan, A., and Trienekens, H. (1987). Vehicle routing with time windows. *Operations Research*, 35(2):266–274.

Kung, H., Luccio, F., and Preparata, F. (1975). On finding maxima of a set of vectors. *Journal of the ACM*, 22(4):469–476.

Larsen, J. (1999). Parallelization of the gehicle routing problem with time windows. *Ph.D Thesis*, Department of Mathematical Modelling, *Technical Report*, University of Denmark.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research*, 23:377–386.

Nemhauser, G. and Wolsey, L. (1988). Integer and Combinatorial Optimization. Wiley, New York.

Powell, W. and Chen, Z. (1998). A generalized threshold algorithm for the shortest path problem with time windows. In: *DIMACS Series in*

Discrete Mathematics and Theoretical Computer Science (P. Pardalos and D. Du, eds.), pp. 303–318. American Mathematical Society.

Rousseau, L.-M., Focacci, F., Gendreau, M., and Pesant, G. (2003). Solving VRPTWs with constraint programming based column generation. *Publication CRT*-2003-10, Center for Research on Transportation, Université de Montréal, Canada.

Ryan, D. and Foster, B. (1981). An integer programming approach to scheduling. In: *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* (A. Wren, ed.), pp. 269–280. North-Holland, Amsterdam.

Savelsbergh, M. and Sol, M. (1998). Drive: Dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):254–265.

Vance, P., Barnhart, C., Johnson, E., and Nemhauser, G. (1997). Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research*, 45(2):188–200.

Villeneuve, D. and Desaulniers, G. (2000). The shortest path problem with forbidden paths. *Les Cahiers du GERAD* G-2000-41, HEC, Montréal, Canada. Forthcoming in: *European Journal of Operational Research.*

Xu, H., Chen, Z., Rajagopal, S., and Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, 37(3):347–364.