The path programming problem and a partial path relaxation

Twan Dollevoet, Diego Pecin, Remy Spliet
Erasmus University Rotterdam
Econometric Institute
dollevoet@ese.eur.nl galindopecin@ese.eur.nl spliet@ese.eur.nl

Econometric Institute Report Series: EI-2020-04

Abstract

We introduce the class of path programming problems, which can be used to model many known optimization problems. A path programming problem can be formulated as a binary programming problem, for which the pricing problem can be modeled as a shortest path problem with resource constraints when column generation is used to solve its linear programming relaxation. Many optimization problems found in the literature belong to this class. We provide a framework for obtaining a partial path relaxation of a path programming problem. Like traditional path relaxations, the partial path relaxation allows the computational complexity of the pricing problem to be reduced, at the expense of a weaker linear programming bound. We demonstrate the versatility of this framework by providing different examples of partial path relaxations for a crew scheduling problem and vehicle routing problem.

Keywords: Path programming, Partial path relaxation, Shortest path problem with resource constraints, Column Generation

1 Introduction

We study a class of binary programming (BP) problems. We specifically consider minimization problems with a very large number of variables, in which case a standard solution approach is to use branch-and-price. That is, a branch-and-bound procedure is employed in which lower bounds are obtained by solving the linear programming (LP) relaxation by means of column generation. In a column generation algorithm, a so-called restricted master problem (RMP) is initialized, which is the LP relaxation in which only a limited number of variables is included. Next, the RMP is solved, followed by solving a so-called pricing problem to identify new variables with negative reduced costs. If no such variables remain, the solution to the RMP is also optimal to the LP relaxation, otherwise the new variables are added to the RMP and we iterate.

In this paper, we specifically focus on a class of BP problems for which the pricing problem can be modeled as a shortest path problem with resource constraints (SPPRC).

The SPPRC was introduced in the Ph.D. thesis [15] of Desrochers in 1986. There exist many applications in which the pricing problem is indeed modeled as an SPPRC, for instance in crew scheduling and vehicle routing [22]. Examples include [3, 4, 6–8, 10, 13, 14, 19, 24]. We impose additional limitations on the class of BP problems, and refer to the resulting class of problems as the path programming (PP) problem. We believe that the features of PP define the case in which it is useful to model the pricing problem of a BP as an SPPRC.

Dror showed that the SPPRC is strongly NP-hard [18]. This justifies the often observed behavior that column generation algorithms spend most of the computation time on solving the pricing problem. As a result, much effort has been spent on acceleration methods such as pricing heuristics.

Of course there are special cases of the SPPRC that are not strongly NP-hard. For example, the shortest path problem without resources might be solved in polynomial time. If all edge weights are positive then the classical Dijkstra algorithm can be used, while if edge weights might be negative but the graph does not contain negative cost cycles then the classical Bellman-Ford algorithm could be used. Furthermore, the shortest walk problem with a single capacity constraint, i.e., load is the single resource and the total load on the walk may not exceed some capacity, is NP-hard, although a pseudo-polynomial time algorithm exists [19]. Here, we distinguish between paths and walks, where a path is a walk that does not contain cycles. In the literature, a path is sometimes also referred to as a simple path or an elementary path to highlight the fact that it does not contain cycles.

To capitalize on the fact that some special cases of the SPPRC are easier to solve, researchers have considered relaxations of their pricing problems. Predominantly, path relaxations are often used in which the elementarity constraint of the SPPRC is relaxed, see for instance [16]. Since in many cases the BP formulation prohibits the selection of variables that correspond to paths that contain cycles, the inclusion of such variables does not alter the optimal integer solution value of the formulation. As a result, the pricing problem becomes easier to solve, although the LP bounds become weaker. Examples include not only allowing any walk [12], but also eliminating k-cycles [12, 22] or ng-path relaxation [5].

In this paper, we suggest an alternative type of relaxation, which we refer to as *partial path relaxation*. The idea is to represent every path in terms of partial paths, and replace them in the formulation of the PP problem, which requires the introduction of additional constraints. The effect of a carefully chosen partial path relaxation will be that the integer optimal solution value does not change, the LP bounds are weaker and the pricing problem is easier to solve. Our work builds on the recent paper by Dollevoet et al. [17] in which a new formulation is presented specifically for the capacitated vehicle routing problem (CVRP), which is based on partial paths of a fixed number of arcs.

We describe a generalized version of the SPPRC, which we refer to as the GSPPRC. Its purpose is to show the general applicability of PP problems and the partial path relaxation.

Nevertheless, the GSPPRC is more general than required in many applications and in the literature a classical SPPRC is usually encountered. For the classical SPPRC we are able to show that some of its properties provide useful benefits for the partial path relaxation. Primarily, we show that it is always possible to reformulate a PP formulation in such a way that the pricing problem becomes polynomially solvable.

Our contribution can be summarized as follows. We generalize the idea presented in [17] to cover all PP problems and not just the CVRP. Moreover, we provide a framework that allows partial paths defined differently than by simply limiting the total number of arcs used. The result is a versatile partial path relaxation. We demonstrate its versatility by applying the partial path relaxation to a crew scheduling problem and a vehicle routing problem.

This paper is organized as follows. In Section 2, we provide the GSPPRC, which we use in Section 3 to define the class of PP problems. There we also provide a binary programming formulation of PP. In Section 4, we introduce partial resource paths which we use in a reformulation of PP to provide a partial path relaxation. This relaxation is valid for any choice of the set of partial resource paths, as long as it is representative of the PP problem. In Section 5, we show that properties of the classical SPPRC provide useful benefits for the partial path relaxation. Here we also provide generic examples of partial path definitions and we show that a reformulation of the PP formulation always exists with a polynomially solvable pricing problem, even when the pricing problem of the PP formulation is NP-hard. We conclude with two examples in which we apply the partial path relaxation. In Section 6, we provide a non-generic partial path relaxation for a crew scheduling problem, which ensures that the resulting pricing problem is polynomially solvable. In Section 7, we compare two different partial path relaxations of the CVRP. Finally, we provide some concluding remarks in Section 8.

2 Generalized shortest path problem with resource constraints

We provide a more general definition of the SPPRC than is commonly done. This allows us to demonstrate the wide applicability of the partial path relaxation.

Consider a directed graph $G = (V, A)$. The vertex set V consists of a source s, a sink t and remaining vertices $V' = V \backslash \{s, t\}$. Note that our use of the word graph includes both simple graphs and multigraphs. If an arc starts at $u \in V$ and ends at $v \in V$, we refer to u as the tail of a, and v as the head of a.

Let R be a set of resources. A *resource vector* is a vector $y \in \mathbb{R}^{|R|}$ such that y_r denotes the amount of resource $r \in R$. We use resource vectors in particular to indicate the amounts of resources at each vertex of a path in G. Denoting the power set of $\mathbb{R}^{|R|}$ by $\mathcal{P}\left(\mathbb{R}^{|R|}\right)$, let $F^a : \mathbb{R}^{|R|} \to \mathcal{P}\left(\mathbb{R}^{|R|}\right)$ be the resource extension function over arc $a \in A$. Given a resource vector y at the tail of arc $a \in A$, the resource extension function F^a

provides the possibly empty domain $F^a(y)$ for a resource vector at the head of arc a. Moreover, let S be the domain of a resource vector at the source s.

This definition of resource extension functions allows for the modeling of many types of resource constraints. An important example of resource extension functions models load. Consider the case of a simple graph and suppose there is a single resource: non-negative load, which is increased by q_v at every vertex v along the path, with an overall capacity limit Q. In this case, $S = \{0\}$ and $F^{(u,v)}(y) = \{y + q_v\} \cap [0, Q]$ for all arcs $(u, v) \in A$. Another important example models time. Again consider a simple graph and suppose there is a single resource: non-negative time, which is increased by t_{uv} at every arc (u, v) along the path. There are time windows $[a_v, b_v]$ at every vertex v along the path, which force you to wait if you arrive before a_v, and disallow you to arrive after b_v. In this case, $S = [a_s, b_s]$ and $F^{(u,v)}(y) = \{\max\{y + t_{uv}, a_v\}\} \cap [a_v, b_v]$.

With *resource path*, we refer to an s, t-path P in G and corresponding resource vectors for every vertex on the path. The resource vector at the source should lie in S, and the resource vectors at each subsequent position should be feasible with respect to the resource extension functions. In particular, for resource vectors y^u and y^v corresponding to the tail u and head v of arc a on a resource path respectively, this means that $y^v \in F^a(y^u)$. We denote by $\mathbb{P}$ the set of all such resource paths. If for some path P there do not exist feasible resource vectors then the path is called resource-infeasible, otherwise it is called resource-feasible.

With each arc $a \in A$, we associate a cost function $C^a : \mathbb{R} \times \mathbb{R}^{|R|} \to \mathbb{R}$. For a current cost c and resource vector y at the tail of arc a, $C^a(c, y)$ provides the new cost after traversing arc a. The cost is initialized at C^s. Note that any vertex cost can be incorporated at the arcs starting at that vertex, except for any cost at the sink. Therefore, we define the cost of a resource path as the cost at the head of the last arc on the path, plus a cost $C^t(y)$ that depends on the resource vector y at the sink t. Note that in many applications, C^t is the zero function. The most common example of cost functions models the accumulation of constant arc costs c_a for each arc $a \in A$. In this case $C^s = 0$, $C^t = 0$, and $C^a(c, y) = c + c_a$. However, like in [20], one might for instance additionally model linear vertex costs with coefficients $\lambda \in \mathbb{R}^{|R|}$ corresponding to the resources, in which case $C^s = 0$, $C^t(y) = \lambda^T y$, and $C^a(c, y) = c + c_a + \lambda^T y$.

The generalized SPPRC (GSPPRC) is the problem of finding a least cost resource path in G from source to sink. Note that the generalized shortest walk problem with resource constraints is a relaxed version of the GSPPRC in which cycles are allowed. It is a common observation that the SPPRC can be modeled as an SWPRC by including a resource for every vertex $v \in V'$ indicating whether v has been visited or not. This observation also holds for the GSPPRC.

3 Path programming problem

We define the class of path programming (PP) problems, by imposing limitations on BP. The class PP contains all BP problems for which, when solved with column generation, the pricing problem can be modeled as a GSPPRC on a graph $G = (V, A)$. Denote by x_p the binary decision variable of BP for $p \in \mathbb{P}$. Here, $\mathbb{P}$ corresponds to the set of resource paths in G, although technically it is sufficient for $\mathbb{P}$ to contain only the optimal resource paths for any pricing problem that might be encountered. We interpret x_p as indicator of whether or not resource path p is selected, and say that a vertex $v \in V$ is visited if a resource path p is selected on which v occurs. We further limit BP by assuming that every vertex $v \in V'$ is visited at most once in any optimal solution. The resulting class of problems is PP.

In Section 3.1, we illustrate that the limitations are merely for ease of exposition. We show that the limitations do not actually exclude any of the instances of BP, showing that PP and BP are equal. Nonetheless, we introduce these limitations to provide a new perspective on BP problems, which help identify those cases for which the techniques presented in this paper are useful. In Section 3.2, we provide a binary programming formulation of PP.

3.1 Restrictiveness of path programming problems

Next, we show that the limitations, which we impose on BP to construct PP, are not restrictive at all. Furthermore, we demonstrate how some BP problems can be modeled as a PP problem, even though it might not seem so at first glance.

Proposition 1. *PP=BP.*

Proof. To see that any instance of BP is included in PP, construct a graph $G_{BP} = (V, A)$ as follows. Let $V = \{s, t\} \cup V'$, where s is a source, t is a sink and additionally V' contains a vertex for every variable of BP. Furthermore, let $A = \{(s, v) : v \in V'\} \cup \{(v, t) : v \in V'\}$, connecting all vertices corresponding to variables to the source and sink. Represent the reduced cost of the variable corresponding to $v \in V'$ in the graph, for instance by assigning the reduced cost to arc (s, v) and cost 0 to the arc (v, t). No resources are required and $C^s = C^t = 0$. This graph G_{BP} allows us to model the pricing problem of an arbitrary instance of BP as a GSPPRC. Clearly, every variable corresponds to a path in G_{BP} and in every feasible solution of BP all vertices in V' are visited at most once. This shows that each instance of BP is included in PP. Since by definition PP is included in BP, we conclude that they are equal. □

Obviously, using G_{BP} to model the pricing problem of BP problems is not very helpful. Solving the corresponding GSPPRC on G_{BP} is algorithmically not that different from what is done in the traditional simplex method, as it amounts to enumerating all variables and evaluating their reduced costs. In particular, such pricing problems render the technique

suggested in this paper moot, and no computational gains should be expected. The reason for this is that we do not think this construction captures the essence of being able to represent a variable as a resource path. For instance, no feasible paths share any vertex in V'. Successful column generation algorithms with a GSPPRC as pricing problem, have in common that the pricing problem is modeled using a relatively small graph, in the sense that the number of variables is typically exponential in the number of vertices and arcs.

Next, let us comment on how to deal with seeming limitations. First, in the definition of PP we have implicitly assumed that the pricing problem is modeled as a single GSPPRC, meaning there is a single source and single sink. Of course there are applications in which the pricing problem decomposes into multiple GSPPRCs, potentially each with their own source and sink. For instance, in the case of heterogeneous fleet VRP [10] a GSPPRC is introduced for every vehicle, for the multi-depot VRP a GSPPRC is introduced for every depot [13] and for the time window assignment VRP a GSPPRC is introduced for every demand scenario [26]. It is not a new insight that these multiple GSPPRCs can be captured in one GSPPRC, where the graphs of the various GSPPRCs are merged into one with an artificial source and sink. Of course in practice, there are usually good reasons to keep the GSPPRCs, and their corresponding types of variables, separate. In that case, the partial path relaxation presented in this paper could best be applied to each type of variables separately. Also note that the class PP is straightforwardly extended to programming problems including continuous or integer variables.

Furthermore, we assume that each vertex $v \in V'$ is visited at most once. In many applications, the BP includes the set partitioning constraints, which imply this. Furthermore, observe that this assumption still allows us to model a BP problem in which a vertex is visited multiple times, as long as an upper bound $\overline{v}$ on the number of visits is known for all $v \in V'$. In that case, we could introduce $\overline{v}$ copies of v. If necessary, any modeling features corresponding to multiple visits of v might be handled through resources in the GSPPRC, or even through the direct inclusion of additional constraints in a BP formulation. In practice, instance specific characteristics might allow the upper bound $\overline{v}$ to be respected without resorting to a cumbersome process of including copies of v.

3.2 Binary programming formulation

We present a binary programming formulation of PP. Let c_p be the cost coefficient of x_p for $p \in \mathbb{P}$. Let a_{vp} be an indicator of whether resource path p visits vertex $v \in V$ and for each additional constraint $h \in H$ with right hand side value b^h let d_p^h be the corresponding constraint coefficient of p. The binary programming formulation of the PP problem is

$$\text{(F-PP)} \quad \min \sum_{p \in \mathbb{P}} c_p x_p \tag{1}$$

$$\sum_{p \in \mathbb{P}} a_{vp} x_p \leq 1 \qquad \forall v \in V' \tag{2}$$

$$\sum_{p \in \mathbb{P}} d_p^h x_p \leq b^h \qquad \forall h \in H \tag{3}$$

$$x_p \in \{0,1\} \qquad \forall p \in \mathbb{P}. \tag{4}$$

For convenience, we explicitly include (2). Although it highlights that every vertex $v \in V'$ is visited at most once, it might not be necessary to include them, specifically if (3) already enforces this implicitly. We assume that the pricing problem of the LP relaxation of F-PP can be modeled as a GSPPRC with vertex set $\{s,t\} \cup V'$ where s and t are the source and sink respectively.

4 Partial path relaxation

In Section 4.1, we define partial resource paths and use them to reformulate the path programming problem. In Section 4.2, we provide potential refinements of the partial path reformulation. Finally, in Section 4.3 we comment on the LP bound and pricing problem of the partial path relaxation corresponding to the partial path reformulation.

4.1 Partial path reformulation

We define a *partial resource path* as a path P in G that does not necessarily start at the source and end at the sink, and corresponding resource vectors for every vertex on the path. Also for partial resource paths, the resource vectors should be feasible with respect to the resource extension functions at each subsequent position on the path. This requires us to define the set of feasible resource vectors at the first vertex on a partial resource path, taking the role that S plays for s. Denoting such a set by S^v for $v \in V'$, it suffices to define $S^v = \mathbb{R}^{|R|}$ for all $v \in V'$. However, one could reduce the set of partial resource paths by for instance defining $S^v = \bigcup_{p \in \mathbb{P}: v \in P(p)} y_p^v$, where we denote by y_p^v the resource vector at vertex v and by $P(p)$ the path of resource path $p \in \mathbb{P}$.

Denote by $\overline{\mathbb{P}}$ a collection of partial resource paths. Two partial resource paths $\overline{p}_1, \overline{p}_2 \in \overline{\mathbb{P}}$ can be concatenated if $\overline{p}_1$ ends at the same vertex $v \in V'$ as where $\overline{p}_2$ starts, and the corresponding resource vectors at v are the same. We say that $\overline{\mathbb{P}}$ *represents* $\mathbb{P}$, if every resource path in $\mathbb{P}$ can be constructed by concatenating partial resource paths in $\overline{\mathbb{P}}$.

Next, we provide a mixed binary linear programming formulation of PP that makes use of variables corresponding to the partial resource paths $\overline{\mathbb{P}}$ instead of resource paths $\mathbb{P}$. It is a reformulation of F-PP, in which we concatenate partial resource paths. We take care that any concatenation exclusively yields a resource path, such that the reformulation

is valid for PP.

We introduce additional parameters to incorporate $\overline{\mathbb{P}}$ in F-PP. Consider a partial resource path $\overline{p} \in \overline{\mathbb{P}}$ and denote by $P(\overline{p})$ the corresponding path of $\overline{p}$. Moreover, let $Y \in \mathbb{R}^{|R|\times|V|}$ be a corresponding resource matrix, such that Y_{rv} is the amount of resource $r \in R$ at vertex $v \in V$, which is zero if v is not visited on $P(\overline{p})$.

Let the parameter $\overline{w}_{v\overline{p}}$ be 1 if $v \in V'$ is the first vertex on $P(\overline{p})$, -1 if it is the last vertex on $P(\overline{p})$, and 0 otherwise. Furthermore, for every resource $r \in R$, let $\overline{y}^v_{\overline{p}r}$ be Y_{rv} if $v \in V'$ is the first vertex on $P(\overline{p})$, $-Y_{rv}$ if it is the last vertex on $P(\overline{p})$, and 0 otherwise. Finally, we introduce a new parameter $\overline{d}^h_{\overline{p}}$ for each partial resource path $\overline{p} \in \overline{\mathbb{P}}$, such that for every decomposition of $p \in \mathbb{P}$ into $p = (\overline{p}(1), \ldots, \overline{p}(l))$, it holds that $d^h_p = \sum_{i=1}^{l} \overline{d}^h_{\overline{p}(i)}$. Note that conceptually this can always be done, since any information on the resource path $p \in \mathbb{P}$ required to determine the value d^h_p could be added as a resource in the representation of a partial resource path. Nonetheless, in many applications, these constraint coefficients are easily decomposed over partial paths. This is also the case for Constraints (2): We do so by defining $\overline{a}_{v\overline{p}}$ to indicate whether $v \in V'$ is visited on partial resource path $\overline{p} \in \overline{\mathbb{P}}$ and v is not the first vertex on the partial resource path. Finally, we similarly introduce the new parameter $\overline{c}_{\overline{p}}$ for $\overline{p} \in \overline{\mathbb{P}}$.

Furthermore, we introduce new continuous variables z_v for $v \in V$. For $v \in V'$, z_v can be interpreted as the position of v on its path. For notational convenience, let $\overline{\mathbb{P}}(u, v)$ be the set of partial resource paths which visit u and v successively. We can now reformulate the PP problem as follows.

$$\left(\overline{\text{F-PP}}\right) \quad \min \sum_{\overline{p}\in\overline{\mathbb{P}}} \overline{c}_{\overline{p}} x_{\overline{p}} \tag{5}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}} \overline{a}_{v\overline{p}} x_{\overline{p}} \leq 1 \qquad \forall v \in V' \tag{6}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}} \overline{d}^h_{\overline{p}} x_{\overline{p}} \leq b^h \qquad \forall h \in H \tag{7}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}} \overline{w}_{v\overline{p}} x_{\overline{p}} = 0 \qquad \forall v \in V' \tag{8}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}} \overline{y}^v_{\overline{p}r} x_{\overline{p}} = 0 \qquad \forall v \in V',\ \forall r \in R \tag{9}$$

$$z_u + (|V'|+1) \sum_{\overline{p}\in\overline{\mathbb{P}}(u,v)} x_{\overline{p}} \leq z_v + |V'| \qquad \forall u, v \in V \tag{10}$$

$$x_{\overline{p}} \in \{0, 1\} \qquad \forall \overline{p} \in \overline{\mathbb{P}} \tag{11}$$

$$z_u \geq 0 \qquad \forall u \in V. \tag{12}$$

Here, (5)-(7) and (11) are the obvious counterparts of (1)-(4) respectively. Constraints (8) ensure that a partial resource path can only be selected that ends at a specific vertex if also a partial resource path is selected that starts at that vertex. Moreover, (9) en-

sure that the resources on these partial resource paths also match. Constraints (10) are subtour elimination constraints which ensure that the concatenation of partial resource paths can never yield cycles in any integer solution. Finally, (11) and (12) define the ranges of the decision variables.

We note that the parameters of each variable depend solely on the resource vectors of the first and last node of the corresponding partial resource path. Thus, two different partial resource paths $\bar{p}$ and $\bar{p}'$ might give rise to variables $x_{\bar{p}}$ and $x_{\bar{p}'}$ that are indistinguishable. In that case, only one of those variables needs to be included. We discuss this issue in more detail in Section 5.2.

The next proposition states that $\overline{\text{F-PP}}$ is a valid reformulation of F-PP.

Proposition 2. *If $\overline{\mathbb{P}}$ represents $\mathbb{P}$ then the optimal objective value of $\overline{\text{F-PP}}$ equals that of F-PP.*

Proof. First observe that any feasible solution to $\overline{\text{F-PP}}$ consists of concatenations of partial resource paths, due to (8) and (9). Given that $\overline{\mathbb{P}}$ represents $\mathbb{P}$, all resource paths can be obtained using an appropriate concatenation of partial resource paths. Moreover, no such concatenation of partial resource paths which results in a resource path $p \in \mathbb{P}$ is prohibited by the formulation (6)-(11). Finally, the corresponding objective values are the same by definition of $c_{\overline{p}}$ for $\overline{p} \in \overline{\mathbb{P}}$. Hence any feasible solution to F-PP corresponds to a feasible solution to $\overline{\text{F-PP}}$ with the same objective value. To conclude that F-PP and $\overline{\text{F-PP}}$ have the same objective value, we show that (6)-(11) are sufficient to ensure that any feasible concatenation of partial resource paths corresponds to a resource path $p \in \mathbb{P}$. Consider a vertex $v \in V'$. Recall that $\overline{a}_{v\overline{p}} = 1$ if v is included in $\overline{p}$ as an intermediate or the final vertex. By (6) and this definition, $\overline{a}_{v\overline{p}} = 1$ for at most one selected partial resource path $\overline{p}$, such that $v \in V'$ is either an intermediate or final vertex on $\overline{p}$. By (8), if v is visited as intermediate vertex, it cannot be visited as first vertex on any other selected partial resource path. If otherwise v is visited as final vertex, by (8) it is also visited as first vertex by exactly one selected partial resource path. Hence, for every vertex $v \in V'$, if a solution arrives at v then v is also departed from. It follows that when considering the concatenation of partial resource paths, it must hold that vertex v is visited on a path from source to sink, or on a cycle. However, by (10) cycles cannot occur. Therefore, every visited vertex $v \in V'$ is contained in a path from source to sink, which by (9) is a resource path. We conclude that F-PP and $\overline{\text{F-PP}}$ have the same optimal objective value. □

4.2 Refinements

We include the subtour elimination constraints in $\overline{\text{F-PP}}$ by introducing $|V|$ variables and $|A|$ constraints (10). Other ways of including subtour elimination constraints are available in the literature. For instance, the following can be used if all vertices must be visited in an optimal solution. Denote by $P(\overline{p})$ the path corresponding to $\overline{p} \in \overline{\mathbb{P}}$ and by $\delta^+(S) \subseteq A$ the arcs going out of $S \subseteq V$. The subtour elimination constraints

$$\sum_{\overline{p}\in\overline{\mathbb{P}}:P(\overline{p})\cap\delta^+(S)\neq\emptyset} x_{\overline{p}} \geq 1 \quad \forall S \subseteq V\backslash\{t\}$$

are a well-known alternative to prevent cycles. This alternative does not require introducing additional variables, and yields a stronger LP bound. However, it requires the introduction of an exponential number of constraints and can only be used if all vertices must be visited.

Subtour elimination constraints actually need not be explicitly included in the formulation. To see this, we define monotonic resources. Denote by $v(a)$ the tail of arc $a \in A$. We say that a resource $r \in R$ is monotonically increasing if for all $a \in A$ and $y \in S^{v(a)}$ it holds that $y_r < \min\{y'_r : y' \in F^a(y)\}$. Similarly, a resource is monotonically decreasing if $y_r > \max\{y'_r : y' \in F^a(y)\}$. If a resource is either monotonically increasing or decreasing, we say it is monotonic. If at least one resource $r \in R$ is monotonic, $\overline{\text{F-PP}}$ remains valid when removing (10) and the variables z, since cycles cannot occur in this case due to (9) for r.

Important examples of monotonic resources that are often encountered in the scientific literature are load and time. Moreover, even in the absence of monotonic resources we can omit (10). This is achieved by deliberately introducing an artificial monotonic resource, although this does require the addition of constraints to (9) for the new resource.

4.3 Partial path relaxation

We refer to the LP relaxation of $\overline{\text{F-PP}}$ as a partial path relaxation. It is not only a relaxation of $\overline{\text{F-PP}}$, but also of F-PP. Denote by z(F-PP) the LP bound of F-PP, and by $z(\overline{\text{F-PP}})$ that of $\overline{\text{F-PP}}$. The LP bound corresponding to the partial path relaxation is weaker.

Proposition 3. *If $\overline{\mathbb{P}}$ represents $\mathbb{P}$ then $z(\overline{\text{F-PP}}) \leq z$(F-PP).*

Proof. Any solution to the LP relaxation of F-PP consists of a, possibly fractional, selection of resource paths in $\mathbb{P}$. Each of these resource paths $p \in \mathbb{P}$ can be represented by a concatenation of partial resource paths from $\overline{\mathbb{P}}$. Selecting these partial resource paths with the same value as with which p is selected, yields a feasible solution to the LP relaxation of $\overline{\text{F-PP}}$ with the same objective value. Note that a partial resource paths may appear more than once, in which case it is selected with the corresponding cumulative value. We conclude that $z(\overline{\text{F-PP}}) \leq z$(F-PP). □

The LP bound of $\overline{\text{F-PP}}$ might be strictly weaker, although this is dependent on the type of partial paths that is considered. A reason for this, for example, is that a fractional solution allows for a convex combination of several partial resource paths all ending at the same vertex $v \in V'$. The resulting amount of resources at v might not be attainable using only resource paths. The corresponding increase in the feasible region could yield a decreased LP bound. In a similar way, the concatenation of partial resource paths in a

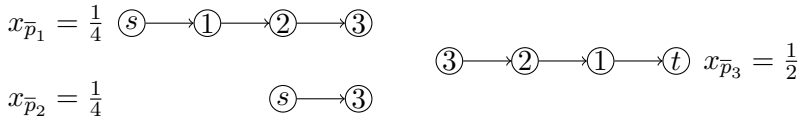

Figure 1: Two reasons for weaker bounds in the partial path relaxation

fractional solution can yield cycles that are unattainable using only resource paths, also yielding a decreased LP bound.

Both of these aspects are illustrated in Figure 1. Consider a PP problem in which all elementary s, t-paths of at most six nodes are allowed. In the pricing problem, we can model this by introducing one resource for the number of nodes on a path. Furthermore, assume that in the partial path relaxation, all partial paths of at most four nodes are allowed. The figure depicts part of a feasible solution of the partial path relaxation. Note that the concatenation of the upper partial paths leads to a full path containing seven nodes, and including cycles. This full path would not be possible in the LP-relaxation of the path programming problem, but can be constructed in the LP-relaxation of the partial path relaxation.

In general, the effect of the partial path relaxation is similar to that of the standard path relaxations, like ng-path relaxation [5]. For standard path relaxations, the set of resource paths is extended by including certain walks, that is, non-elementary paths in which vertices can be visited multiple times. By doing so, cycles are allowed, where the type of cycles depends on the type of path relaxation. Now, a reformulation is obtained by simply replacing the original set of resource paths by the extended set that includes walks. This can be done if the formulation enforces that every vertex is visited at most once, in which case the integer optimal solution is not affected. Hence, this is exclusively applicable to the problems included in PP. The result of standard path relaxations is that the LP relaxation becomes weaker while the pricing problem typically becomes easier to solve. These effects can also be observed for partial path relaxation. Next, we comment in broad terms on how the pricing problem of a partial path relaxation compares to that of F-PP.

The pricing problem of $\overline{\text{F-PP}}$ is to find a path that does not necessarily start at the source and end at the sink. Furthermore, the dual multipliers of (8)-(10) are part of the reduced cost. Moreover, the choice of which type of partial paths to consider obviously affects the structure of the pricing problem.

Firstly, that a path need not start nor end at the sink does not necessarily affect the complexity of the pricing problem. Indeed, one might decompose the pricing problem into $O(|V|^2)$ problems, one pricing problem for each start and end vertex. If the complexity of the pricing problem is polynomial and not dependent on the start and end vertex, the pricing problem remains polynomially solvable. Moreover, with the advent of parallel computing, one can solve the decomposed pricing problems in parallel if enough cores are

available.

Secondly, when the pricing problems are decomposed this way, the incorporation of Constraints (8) does not affect the complexity as well. The contribution of the corresponding dual multipliers to the reduced cost is constant when the start and end location of the partial path is fixed.

Thirdly, however, the inclusion of (9) could potentially increase the complexity of the pricing problem. Due to (9), the initial and final resource vectors of a partial path represented by $\overline{y}^v_{\overline{p}r}$ are included in the reduced costs, and become decision variables in the pricing problem that require optimization. The exception obviously being when there are no resources to consider.

Fourthly, note that the dual multipliers corresponding to constraints (10) can be added to the costs of an arc in the GSPPRC. Although the addition of a multiplier for arcs might affect the complexity of the pricing problem, this is not common and in practice it typically does not affect algorithm design.

Finally, the choice of the type of partial paths could potentially decrease the complexity of the pricing problem. An example of this is found in [17], where a partial path relaxation for CVRP is used for which the pricing problem is polynomially solvable, while traditionally the pricing problem is modeled as an NP-hard problem. Even if the pricing problem of $\overline{\text{F-PP}}$ is in the same complexity class as that of F-PP, there might be other computational gains.

5 Specific partial path relaxations

We have introduced the generalized version of the SPPRC, GSPPRC, to convey the wide applicability of the partial path relaxation. Note that the main idea behind the partial path relaxation is to reduce the computational effort of solving the pricing problem of $\overline{\text{F-PP}}$ compared to that of F-PP. However, in Section 5.1 we argue that due to the generality of GSPPRC the pricing problem of $\overline{\text{F-PP}}$ is in general still very difficult.

Fortunately, the GSPPRC is perhaps unnecessarily general for many applications. Researchers have mostly worked with the SPPRC instead. In Section 5.2, we discuss the SPPRC with so-called 'classical' resources as encountered in the literature, and derive additional insights in the context of the partial path relaxation. The complexity of the pricing problem is also dependent on the definition of the partial paths, and we provide some generic examples of this in Section 5.3. Finally, in Section 5.4 we comment on the difference in complexity of the pricing problems of F-PP and that of its partial path relaxation.

5.1 GSPPRC is difficult

Propositions 1 and 2 illustrate the generality of our methodology. Indeed, the partial path relaxation can be applied to every binary program for which the pricing problem can be

formulated as a GSPPRC as defined in Section 2. However, due to its generality, the GSPPRC is a very difficult optimization problem. Even determining whether a path P is resource-feasible is NP-hard.

Proposition 4. *Determining for an instance of the GSPPRC whether a given path P is resource-feasible is NP-hard.*

Proof. Let an instance of PARTITION be given. This instance contains n positive integers a_i, for $i \in \{1, \ldots, n\}$. Define a directed graph $G = (V, A)$ where $V = \{1, \ldots, n\}$ and $A = \{(i, j) : j > i\}$. Consider one resource and define $s = 1$, $t = n$, and $S = \{0, a_1\}$. Moreover, define

$$F^{(i,j)}(y) = \begin{cases} \{y, y + a_j\} \cap \{M\} & \text{if } j = n, \\ \{y, y + a_j\} & \text{otherwise,} \end{cases}$$

where $M = \frac{1}{2} \sum_{i=1}^{n} a_i$. Then, $P = (1, 2, \ldots, n)$ is resource-feasible if and only if the instance of PARTITION is a YES-instance. Hence, determining whether P is resource-feasible is NP-hard. □

As a result, not only is the pricing problem of F-PP very difficult, the pricing problem of $\overline{\text{F-PP}}$ is also very difficult in general.

5.2 Classical resources

Irnich and Desaulniers describe a less general SPPRC than the GSPPRC, which they refer to as the 'classical' SPPRC in [21]. It has the classical resource extension functions of the form (in our notation)

$$F^a(y) = [y + t^a, \infty) \cap [a^v, b^v]$$

where v is the head of a and $t^a, a^v, b^v \in \mathbb{R}^{|R|}$. Furthermore $S = [a^s, b^s]$. In this case, the resource vectors are separable by resource. For the remainder of this paper, we refer to this 'classical' SPPRC simply as SPPRC.

Contrary to the case of GSPPRC, for an SPPRC it is easy to verify whether a given path P is resource-feasible. Let P consist of the arcs $(a(1), \ldots, a(k))$, such that the tail of $a(1)$ is the source s and the head of $a(k)$ is the sink. We define $y_r^s = \min\{y_r : y \in S\}$ and, recursively, $y_r^{v(i+1)} = \min\{y_r : y \in F^{a(i)}(y^{v(i)})\}$ with $v(i)$ the tail and $v(i+1)$ the head of $a(i)$, for as long as $F^{a(i)}(y^{v(i)}) \neq \emptyset$ and $i \leq k$. If and only if during the recursion $F^{a(i)}(y^{v(i)}) = \emptyset$ for some $i \in \{1, \ldots, k\}$, the path P is resource-infeasible. Using this recursion to verify whether path P is resource-feasible requires a number of computations that is linear in the length of P.

Next, we focus on the case that the pricing problem of F-PP is an SPPRC and additionally assume constant arc costs, i.e., we assume that $C^a(c, y) = c + c_a$ for all $a \in A$ and that $C^s = C^t = 0$. We show that in this case we only require a limited number of partial resource paths in $\overline{\mathbb{P}}$ for any partial path relaxation. Roughly stated, it turns out that we

only need to consider those partial resource paths with the resource amounts at the first resource as low or as high as possible. This does not only reduce the size of the partial path reformulation $\overline{\text{F-PP}}$ but also means that the complexity of the pricing problem does not suffer much from including (9) in the reformulation. Next, we make this statement more precise.

First note that it is sufficient for the validity of $\overline{\text{F-PP}}$ to define $S^v = [a^v, b^v]$, for all $v \in V$. Observe that when using this definition of S^v and the classical resource extension functions, in the partial path relaxation $\overline{\text{F-PP}}$ we can replace equations (9) by the following inequalities

$$\sum_{\bar{p} \in \overline{\mathbb{P}}} \overline{y}^v_{\bar{p}r} x_{\bar{p}} \geq 0 \quad \forall v \in V',\ \forall r \in R. \tag{13}$$

Consider a partial path P in G consisting of the vertices $(v(1), \ldots, v(k+1))$. Define $Y(P) \subseteq \mathbb{R}^{|R| \times |V|}$ as the set of feasible resource matrices corresponding to partial path P. We say that a resource matrix $Y \in Y(P)$ dominates a resource matrix $\hat{Y} \in Y(P)$ if $\hat{Y}_{rv(1)} \leq Y_{rv(1)}$ and $Y_{rv(k+1)} \leq \hat{Y}_{rv(k+1)}$ for all $r \in R$ and at least one of the inequalities is strict. Any solution to $\overline{\text{F-PP}}$ with (13) instead of (9), in which a partial resource path is selected corresponding to P and $\hat{Y}$ can be changed by replacing this partial resource path with that using P and Y. In particular, the optimal solution value remains the same by the assumption of constant arc costs, and the new solution is feasible. Note specifically that it does not violate (13).

Furthermore, recall from Section 4.1 that two different partial resource paths $\bar{p}$ and $\bar{p}'$ might give rise to variables $x_{\bar{p}}$ and $x_{\bar{p}'}$ that are indistinguishable if the resource vectors at the first and last vertex of $\bar{p}$ and $\bar{p}'$ are equal. In that case, we consider only that partial resource path for which the resource vectors at the intermediate vertices are minimal. We define $\tilde{Y}(P)$ as the set of such minimal non-dominated resource matrices for P.

In the following, we might refer to a singleton as an interval: We consider $[a, a] = \{a\}$ as an interval as well. We have the following proposition.

Proposition 5. *Consider an SPPRC on the graph G and let a partial path P in G be given with arc representation $(a(1), \ldots, a(k))$ and vertex representation $(v(1), \ldots, v(k+1))$. For the set $\tilde{Y}(P)$ of minimal non-dominated resource matrices there are the following three options, where the second and third option are not mutually exclusive.*

1. *The partial path P is resource-infeasible, so that $\tilde{Y}(P) = Y(P) = \emptyset$.*
2. *There is only one minimal non-dominated resource matrix, i.e., $\tilde{Y}(P) = \{Y\}$.*
3. *For all $r \in R$, there is an interval $\left[\tilde{a}_r, \tilde{b}_r\right] \subseteq \mathbb{R}$ such that $Y \in \tilde{Y}(P)$ if and only if $Y_{rv(1)} \in \left[\tilde{a}_r, \tilde{b}_r\right]$ and, for all $r \in R$, $i \in \{1, \ldots, k\}$ it holds that $Y_{rv(i+1)} = Y_{rv(i)} + t_r^{a(i)}$.*

Proof. We provide a proof by induction on the length k of the path P. For paths of length 0 the third option applies, and in particular $\left[\tilde{a}_r, \tilde{b}_r\right] = \left[a_r^{v(1)}, b_r^{v(1)}\right]$. Assume now that

the statement holds for paths of length m. Consider any path P of length $m+1$ and let path P' of length m be obtained by removing the last vertex from P, that is, by removing vertex $v(m+2)$. Hence, P' satisfies one of the three options, which we next consider individually.

If the first option applies to P', then it obviously applies to P as well.

If the second option applies to P', denote by $Y' \in \tilde{Y}(P')$ the only minimal non-dominated resource matrix corresponding to P'. If there is a resource $r \in R$ such that $Y'_{rv(m+1)} + t_r^{a(m+1)} > b_r^{v(m+2)}$, the path P is resource-infeasible and the first option applies to P. Otherwise, consider the resource matrix Y obtained from Y' by setting $Y_{rv(m+2)} = \max\left\{a_r^{v(m+2)}, Y'_{rv(m+1)} + t_r^{a(m+1)}\right\}$ for all $r \in R$. It follows that $Y \in Y(P)$, and moreover that it dominates any other resource matrix in $Y(P)$. Hence $\tilde{Y}(P) = \{Y\}$, which means that the second option applies to P.

Finally, consider the case that the third option applies to P', and let $\left[\tilde{a}'_r, \tilde{b}'_r\right]$ be the interval such that $Y' \in \tilde{Y}(P')$ if and only if $Y'_{rv(1)} \in \left[\tilde{a}'_r, \tilde{b}'_r\right]$ for all $r \in R$. Note that in this case, any non-dominated resource matrix corresponding to P' is a minimal non-dominated resource matrix. We next show that all three options could apply to P, but no other.

The first option applies to P in the following case. Select $Y' \in \tilde{Y}(P')$ such that the resource amount of each resource $r \in R$ is as small as possible at the initial vertex, i.e., $Y'_{rv(1)} = \tilde{a}'_r$. If there is a resource $r \in R$ such that $Y'_{rv(m+1)} + t_r^{a(m+1)} > b_r^{v(m+2)}$, the path P is resource-infeasible and the first option applies to P.

The second option applies to P in the following case. Select $Y' \in \tilde{Y}(P')$ such that the resource amount of each resource $r \in R$ is as high as possible at the initial vertex, i.e., $Y'_{rv(1)} = \tilde{b}'_r$. If for all resources $r \in R$ it holds that $Y'_{rv(m+1)} + t_r^{a(m+1)} \leq a_r^{v(m+2)}$ then there is one dominating resource matrix. Indeed, for the matrix Y obtained from Y' by setting $Y_{rv(m+2)} = a_r^{v(m+2)}$ for all $r \in R$ it holds that $\tilde{Y}(P) = \{Y\}$.

Otherwise the third option must apply to P as we demonstrate next. Define $\tilde{a}_r$ as the smallest amount of resource $r \in R$, for which a non-dominated resource matrix $Y' \in \tilde{Y}(P')$ can be found that can be extended to a non-dominated resource matrix $Y \in \tilde{Y}(P)$. This is achieved as follows.

$$\tilde{a}_r = \min\left\{ x : \begin{array}{l} Y' \in \tilde{Y}(P'), \\ Y'_{rv(1)} = x, \\ Y'_{rv(m+1)} + t_r^{a(m+1)} \in \left[a_r^{v(m+2)}, b_r^{v(m+2)}\right] \end{array} \right\}$$

Similarly, we define the largest such amount $\tilde{b}_r$ as follows.

$$\tilde{b}_r = \max\left\{ x : \begin{array}{l} Y' \in \tilde{Y}(P'), \\ Y'_{rv(1)} = x, \\ Y'_{rv(m+1)} + t_r^{a(m+1)} \in \left[a_r^{v(m+2)}, b_r^{v(m+2)}\right] \end{array} \right\}$$

In this case $\tilde{a}_r$ and $\tilde{b}_r$ are well-defined and form a non-empty interval $\left[\tilde{a}_r, \tilde{b}_r\right]$. Observe

that for $Y' \in \tilde{Y}(P')$ with $Y'_{rv(1)} \in \left[\tilde{a}_r, \tilde{b}_r\right]$ we can construct a $Y \in \tilde{Y}(P)$ from Y' by setting $Y_{rv(m+2)} = Y'_{rv(m+1)} + t_r^{a(m+1)}$ for all $r \in R$. It is easily verified that Y is non-dominated. We conclude that the third option applies to P.

The proposition follows by induction. □

We have already described that in $\overline{\text{F-PP}}$ we can limit the set of variables to correspond to partial paths and minimal non-dominated resources matrices. Using Proposition 5, it follows that we can limit the set of variables even further. To see this, define $Y^*(P) \subseteq \tilde{Y}(P)$ as the set consisting of every minimal non-dominated resource matrix corresponding to a resource-feasible partial path P, such that the amount of each resource at the initial vertex is either minimal or maximal. That is, for $Y \in Y^*(P)$ it holds for every $r \in R$ that $Y_{rv(1)} = \min_{Y' \in \tilde{Y}(P)} Y'_{rv(1)}$ or $Y_{rv(1)} = \max_{Y' \in \tilde{Y}(P)} Y'_{rv(1)}$. It immediately follows from Proposition 5 that every $Y \in \tilde{Y}(P)$ can be written as a convex combination of the resource matrices in $Y^*(P)$.

Corollary 1. *Consider an SPPRC on the graph G and consider a resource-feasible partial path P in G. It holds that* $\text{CONV}\left(Y^*(P)\right) = \tilde{Y}(P)$.

By construction it holds that $|Y^*(P)| \leq 2^{|R|}$, which is a constant for a fixed number of resources. Moreover, it follows from the proof of Proposition 5 that $Y^*(P)$ can be constructed in a number of computations that is linear in the length of P.

We are now able to provide an alternative reformulation of F-PP when the pricing problem is an SPPRC with constant arc costs, which for many cases has less variables than $\overline{\text{F-PP}}$. We refer to this reformulation as $\overline{\text{F-PP}}(SPPRC)$. Denote by $\overline{\mathbb{P}}^* \subseteq \overline{\mathbb{P}}$ the set of partial resource paths P for which it holds that the corresponding resource matrix $Y \in Y^*(P)$. Furthermore, denote by RFP-Paths the set of resource-feasible partial paths. The set RFP-Paths consists of partial paths in G such that $Y(P) \neq \emptyset$, and should not be confused with the set of partial resource paths $\overline{\mathbb{P}}$. Denote by $\overline{\mathbb{P}}^*(P) \subseteq \overline{\mathbb{P}}^*$, all partial resource paths corresponding to partial path $P \in$ RFP-Paths. The alternative reformulation is

$$\left(\overline{\text{F-PP}}(SPPRC)\right) \quad \min \sum_{\overline{p}\in\overline{\mathbb{P}}^*} \overline{c}_{\overline{p}} x_{\overline{p}} \tag{14}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}^*} \overline{a}_{v\overline{p}} x_{\overline{p}} \leq 1 \qquad \forall v \in V' \tag{15}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}^*} \overline{d}^h_{\overline{p}} x_{\overline{p}} = b^h \qquad \forall h \in H \tag{16}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}^*} \overline{w}_{v\overline{p}} x_{\overline{p}} = 0 \qquad \forall v \in V' \tag{17}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}^*} \overline{y}^v_{\overline{p}r} x_{\overline{p}} \geq 0 \qquad \forall v \in V',\ \forall r \in R \tag{18}$$

$$\sum_{\overline{p}\in\overline{\mathbb{P}}^*(P)} x_{\overline{p}} \in \{0,1\} \qquad \forall P \in \text{RFP-Paths} \tag{19}$$

$$x_{\overline{p}} \geq 0 \qquad \forall \overline{p} \in \overline{\mathbb{P}}^*. \tag{20}$$

Here, (14)-(17) are the counterparts of (5)-(8), and (18) is the counterpart of (13) which replaces (9) of $\overline{\text{F-PP}}$ for the case of SPPRC. Finally, (19) and (20) are the new integrality conditions, which allow a convex combination of partial resource paths, as long as the resulting combination selects partial paths binarily.

Observe that $\overline{\text{F-PP}}(SPPRC)$ uses $\mathcal{O}(|\overline{\mathbb{P}}^*|)$ variables as opposed to the potentially infinite number of variables used by $\overline{\text{F-PP}}$. Note that the binarity conditions (11) of $\overline{\text{F-PP}}$ could also have been reduced to only $\mathcal{O}(|\text{RFP-Paths}|)$ binarity conditions using similar constraints as (19). In the scientific literature, however, it is uncommon to enforce binary conditions like (11) or (19) directly anyway. Usually, binarity conditions are enforced using some form of Ryan/Foster branching [25], e.g., binarity of flow on an arc instead of path selection. Finally, note that the LP relaxation of $\overline{\text{F-PP}}(SPPRC)$ is provided by (14)-(18) and (20), which by construction has the same value as the LP bound of $\overline{\text{F-PP}}$.

5.3 Generic partial path definitions

When deciding on the definition of the set of partial resource paths $\overline{\mathbb{P}}$, three things are crucial to consider. First of all, every feasible resource path should be represented by the partial resource paths. Secondly, one should aim for reducing the computational burden of solving the resulting pricing problem. Thirdly, the decrease in LP bound should not offset the gain in computation time for the pricing problem. In particular, the aim is to reduce the computation time of the overall branch-and-price algorithm. Next, we discuss some generic options that can always be applied.

We can define $\overline{\mathbb{P}}$ as the set of all possible partial resource paths in the graph that satisfy a limitation on the length, while setting $S^v = [a^v, b^v]$ for all $v \in V'$. One possibility to do so has been introduced by Dollevoet et al. [17] for the CVRP, but the main idea

applies here as well. We consider partial resource paths of *roughly length* k. Note that not all resource paths might be represented by partial resource paths of length exactly k. To remedy this, in [17] partial resource paths are considered of length exactly k, or at most k if the partial resource path starts at the source. For the sake of exposition, we propose here to consider partial resource paths of length exactly k, or at most k if the partial resource path *ends* at the sink. All resource paths are represented by the set of partial resource paths of roughly length k, and the total number of such partial resource paths for fixed k is polynomial in $|V|$ and $|A|$. Furthermore, an increase in LP bound can be observed when k increases. The following proposition was provided by Dollevoet et al. [17] for the case of the CVRP, but also applies in our more general case. Denote by $z_k(\overline{\text{F-PP}})$ the LP relaxation of $\overline{\text{F-PP}}(SPPRC)$ where all feasible partial resource paths of length exactly k are included, or at most k if it ends at the sink.

Proposition 6. *For any $k, m \in \mathbb{N}_{>0}$ it follows that $z_k(\overline{F\text{-}PP}) \leq z_{km}(\overline{F\text{-}PP})$.*

Proof. Consider an optimal solution x^* to the LP relaxation using partial resource paths of roughly length km. In order to construct a solution x' to $\overline{\text{F-PP}}(SPPRC)$ using partial resource paths of roughly length k, initialize $x'_{\overline{p}} = 0$ for all $\overline{p} \in \overline{\mathbb{P}}$. Any given partial resource path r of roughly length km can be cut into l partial resource paths $\overline{p}(1), \ldots, \overline{p}(l)$ of roughly length k, where l is at most m. We now add x^*_r to $x'_{\overline{p}(i)}$ for all $1 \leq i \leq l$. It is easily verified that x' is a feasible solution using exclusively partial resource paths of roughly length k, while maintaining the same objective value. This shows that $z_k \leq z_{km}$. □

Other straightforward options to limit the path length is by considering partial resource paths of length at most k, or at least k. Note that in the former case the LP bound is $z_1(\overline{\text{F-PP}})$, while in the latter case the LP bound is at most $z_k(\overline{\text{F-PP}})$. Both these options are unlikely to yield computational gains over using partial resource paths of roughly length k.

Another possibility to define partial resource paths is based on the resource matrix. To illustrate this, consider the case where there is a monotonically increasing resource. We refer to the difference between the amount of such a resource between two vertices on a partial resource path as *resource consumption* and say that the resource consumption between the first and last vertex on a path is the total resource consumption. Similar to a limit on the partial path length we set $S^v = [a^v, b^v]$ for all $v \in V'$ and limit resource consumption, although placing a limit on total resource consumption analogous to a limit of roughly length k requires some care.

Observe that in many applications, paths cannot be decomposed in partial paths with total resource consumption exactly some given fixed value q, not even when allowing partial paths ending at the sink with total resource consumption at most q. Instead, we suggest the following. We consider partial paths on which the resource consumption between the first and the semi-last vertex is at most q, while the resource consumption between the

first and the last vertex is strictly larger than q. Furthermore, we consider all partial paths that end at the sink and have a total resource consumption of at most q. Every resource path can be represented using these partial resource paths with *roughly resource consumption* q. In case the resource consumption of every arc is 1, this corresponds to considering partial resource paths of roughly length $k = q + 1$.

Note that for similar reasons as in the case of a limit on length, it does not seem sensible to consider limiting the total resource consumption to at most q, or at least q. Finally note that something similar can of course be done for monotonically decreasing resources.

Although the above examples are generic, application specific definitions of the set of partial resource paths $\overline{\mathbb{P}}$ might provide more computational gains. In Section 6, we provide a case specific example in crew scheduling. An example of partial paths of roughly length k for the CVRP can be found in [17], which we compare to an example of partial path of roughly resource consumption q in Section 7.

5.4 Complexity of the pricing problem

Next we comment on the difference in the complexity of the pricing problems of F-PP and its partial path relaxation.

Proposition 7. *For any problem formulated as F-PP for which the pricing problem is an SPPRC with a fixed number of resources, there exists a partial path reformulation of which the pricing problem is polynomially solvable in the size of the graph corresponding to the SPPRC.*

Proof. Let the set of partial resource paths $\overline{\mathbb{P}}$ consist of all partial resource paths of roughly length k. Consider the partial path reformulation of F-PP, denoted by $\overline{\text{F-PP}}(SPPRC)$. Observe that as stated in Section 5.3, the number of partial resource paths of roughly length k is polynomial in $|V|$ and $|A|$. Therefore, for a fixed number of resources, $\overline{\text{F-PP}}(SPPRC)$ has a polynomial number of variables. Hence, the pricing problem of $\overline{\text{F-PP}}(SPPRC)$ can be solved in polynomial time by enumeration. □

The significance of Proposition 7 is that it also applies in case the pricing problem of F-PP is strongly NP-hard. It has been proven that the SPPRC is strongly NP-hard [18]. In particular, the proof in [18] uses an SPPRC that is used to model the pricing problem of a set partitioning formulation of the vehicle routing problem with time windows. It is noteworthy that this set partitioning formulation coincides with F-PP and the pricing problem of a partial path reformulation is polynomially solvable.

6 A crew scheduling example

Next, we provide an example of an optimization problem that can be modeled as a PP problem of which the pricing problem is commonly modeled as an NP-hard SPPRC. It is

a crew scheduling problem found in the literature. We provide a partial path relaxation other than the generic examples found in Section 5.3. Moreover, we show that the pricing problem of the corresponding reformulation is polynomially solvable.

6.1 Problem description

We consider a crew scheduling problem as described in [9] for which column generation algorithms are used in the literature as well as in practice, see e.g. [1]. Let V' be a set of tasks. Each task $v \in V'$ has a specified start time t_v^S and end time t_v^E. We define $t_v = t_v^E - t_v^S$ as the *work duration* of task v.

An unlimited number of crew members is available for performing the tasks, and each task has to be performed exactly once by one crew member. Let C be a set consisting of pairs of tasks that can be performed consecutively. We note that a necessary, but not sufficient, condition for a pair of tasks (u, v) to be in C, is that the end time of u is strictly prior to the start time of v, i.e., $t_u^E < t_v^S$. A duty is a sequence of tasks that can be performed consecutively by one crew member. Each duty p has a cost coefficient c_p, which consists of costs c_{uv} for all pairs of tasks $(u, v) \in C$ that are performed consecutively in duty p, a cost c_{sv} if $v \in V'$ is the first task in the duty, and a cost c_{vt} if v is the last task in the duty.

Each duty satisfies the following labour regulations. The cumulative work duration of all tasks in a duty may not exceed W. Furthermore, a duty can have no or one break. A break can be had between consecutive tasks in a duty, denote by $C^B \subseteq C$ all pairs of tasks between which a break can be had. We define a *duty length* between two tasks as the difference between the start time of the earliest task and the end time of the latest task. The duty length without a break may not exceed L. If a duty contains a break, the maximum duty length between the first task and the last task before the break is L, as well as the maximum duty length between the first task after the break and the final task. We assume that $L \leq W < 2L$. Denote by $\mathbb{P}$ the set of duties.

The objective is to select a set of duties of minimal cost such that each task is contained in one of the selected duties. For the remainder of this paper, we refer to this specific problem as the crew scheduling problem.

6.2 Path programming formulation

Next, we provide a path programming formulation for the crew scheduling problem. Let a_{vp} indicate whether task $v \in V'$ is included in duty $p \in \mathbb{P}$. Furthermore, the binary variable x_p indicates whether duty $p \in \mathbb{P}$ is selected. A path programming formulation F-PP of the crew scheduling problem is the following.

$$\min \sum_{p \in \mathbb{P}} c_p x_p \tag{21}$$

$$\sum_{p \in \mathbb{P}} a_{vp} x_p = 1 \qquad \forall v \in V' \tag{22}$$

$$x_p \in \{0,1\} \qquad \forall p \in \mathbb{P} \tag{23}$$

This formulation is a common set partitioning formulation for the crew scheduling problem. To classify this problem as a path programming problem, we demonstrate that the pricing problem can be modeled as an SPPRC.

Consider the multigraph $G = (V, A)$, where $V = V' \cup \{s, t\}$. The set of arcs A consists of i) an arc between the source s and v for all $v \in V'$, ii) an arc between $u, v \in V'$ for every $(u, v) \in C$, iii) an arc between v and the sink t for all $v \in V'$, and iv) an additional arc between $u, v \in V'$ for every $(u, v) \in C^B$. We denote the set of arcs of the latter type as $A^B \subseteq A$, and refer to them as break-arcs. Note that, for every pair in C^B, two arcs are included in G. Traversing the arc in A^B corresponds to having a break, while the other corresponds to not having a break. Note that the graph G is acyclic by construction of C.

We define three resources, $R = \{1, 2, 3\}$. The first resource corresponds to the cumulative work duration. Let the initial cumulative work duration be 0, i.e., $S^1 = \{0\}$. For each arc $a \in A$, with head $v \in V'$, we define the resource extension function for this resource as $F_1^a(y) = [y_1 + t_v, \infty) \cap [0, W]$. If the head of a is t then $F_1^a(y) = [y_1, \infty) \cap [0, W]$. This way, traversing an arc increases the cumulative work duration with the work duration of the task at the end of the arc.

The second resource corresponds to the duty length without a break. Let $S^2 = \{0\}$. For each arc $a = (s, v) \in A$, we define the resource extension function for this resource as $F_2^a(y) = [y_2 + t_v, \infty) \cap [0, L]$. If $a = (v, t) \in A$, we define $F_2^a(y) = [y_2, \infty) \cap [0, L]$. For $a \in A^B$, we define $F_2^a(y) = [0, L]$. Finally, for all other arcs $a = (u, v) \in A$, with $(u, v) \in C$, we define $F_2^a(y) = [y_2 + t_v^E - t_u^E, \infty) \cap [0, L]$. This way, the duty length is effectively reset when a break-arc is traversed.

Finally, the third resource is used to indicate whether a break has been had. Let $S^3 = \{0\}$. For each arc $a \in A \backslash A^B$ let $F_3^a(y) = \{y_3\}$. For arc $a \in A^B$ let $F_3^a(y) = [y_3 + 1, \infty) \cap [0, 1]$. This ensures that any resource-feasible s, t-path in G traverses at most one break-arc. We now define $F^a(y) = F_1^a(y) \times F_2^a(y) \times F_3^a(y)$ and $S = S^1 \times S^2 \times S^3$.

We associate a cost to each arc $a \in A$. Finding the least cost resource path in G is an SPPRC, which we denote by SPPRC(1).

Observe that a duty corresponds to a resource path in G. Denote by λ the dual multipliers corresponding to (22), and let $\lambda_t = 0$. We model the pricing problem of (21)-(23) as an SPPRC(1), by associating with each arc $a \in A$, with head $v \in V$, the cost $c_a - \lambda_v$. Finally, note that even though A is acyclic and checking whether a path is resource feasible can be done in polynomial time, still SPPRC(1) is NP-hard.

Proposition 8. *SPPRC(1) is NP-hard.*

Proof. Let an instance of PARTITION be given. This instance contains n positive integers a_i, for $i \in \{1, \ldots, n\}$, we may assume without loss of generality that $\sum_{i=1}^{n} a_i \geq 2$. Consider an instance of SPPRC(1) where $V' = \{1, \ldots, n\}$, and we denote $V = \{0, 1, \ldots, n, n+1, n+2\}$, where 0 and $n+2$ are the source and sink respectively, while $n+1$ represents a long task. Define $M = n + \sum_{i=1}^{n} a_i$. Let $a_{n+1} = M$ and $a_{n+2} = 0$. Furthermore, denote $t_0^E = 0$ and let $t_i^S = t_{i-1}^E + 1$ and $t_i^E = t_i^S + a_i$ for $i \in \{1, \ldots, n+2\}$. Let $A^B = \{(i, n+1) : 1 \leq i \leq n\}$ and $A = \{(i,j) : 0 \leq i < j \leq n+2\} \cup A^B$. Furthermore, let $W = M + \frac{1}{2}\sum_{i=1}^{n} a_i$ and $L = M + 1$, satisfying $L \leq W < 2L$. Observe that to include task $n+1$, a break is required to include more tasks. For $(i,j) \in A$, define the arc cost as $c_{ij} = -a_j$. As a result, any optimal solution includes task $n+1$. The optimal solution value of this instance of the SPPRC(1) is $-W$ if and only if PARTITION is a YES-instance. Hence, SPPRC(1) is NP-hard. □

We remark that SPPRC(1) can be solved in pseudo-polynomial time. This is achieved for instance by dynamic programming in $\mathcal{O}(|A|WL)$, exploiting the fact that the graph is acyclic and hence a topological ordering of the vertices can be obtained. Hence, also the pricing problem of (21)-(23) can be solved in pseudo-polynomial time.

6.3 Partial path relaxation

We reformulate F-PP of the crew scheduling problem as follows. Let $\overline{\mathbb{P}}$ consist of i) partial resource paths starting at the source, not traversing any break-arc, and ii) partial resource paths which do not start at the source, but do end at the sink, of which the first arc is a break-arc. Effectively, this means that $\overline{\mathbb{P}}$ consists of partial paths before or after the start of the break, and in the reformulation, constraints are included to concatenate such partial paths at the start of the break. Note that it is sufficient to set $S^v = [0, W] \times [L] \times \{0\}$ for all $v \in V'$. The partial path reformulation $\overline{\text{F-PP}}$ is

$$\left(\overline{\text{F-PP}}\right) \quad \min \sum_{\overline{p} \in \overline{\mathbb{P}}} \overline{c}_{\overline{p}} x_{\overline{p}} \tag{24}$$

$$\sum_{\overline{p} \in \overline{\mathbb{P}}} \overline{a}_{v\overline{p}} x_{\overline{p}} = 1 \qquad \forall v \in V' \tag{25}$$

$$\sum_{\overline{p} \in \overline{\mathbb{P}}} \overline{w}_{v\overline{p}} x_{\overline{p}} = 0 \qquad \forall v \in V' \tag{26}$$

$$\sum_{\overline{p} \in \overline{\mathbb{P}}} \overline{y}^v_{\overline{p}1} x_{\overline{p}} \geq 0 \qquad \forall v \in V' \tag{27}$$

$$x_{\overline{p}} \in \{0, 1\} \qquad \forall \overline{p} \in \overline{\mathbb{P}}. \tag{28}$$

Observe that the first resource, corresponding to cumulative work duration is monotonic, so no subtour elimination constraints are required. Furthermore, only for this

resource are constraints of the type (27) required. By definition of $\overline{\mathbb{P}}$, the analogue of (27) for the second resource, duty length without a break, is in some sense equivalent with (26), while for the third resource the analogue of (27) are trivially satisfied.

We conclude this example by showing that the pricing problem of the reformulation (24)-(28) is polynomially solvable.

Proposition 9. *The pricing problem of* (24)-(28) *is polynomially solvable.*

Proof. Denote by $\overline{\lambda}$, $\overline{\mu}$ and $\overline{\pi}$ the dual multipliers of (25)-(27) respectively. First consider a partial path $\overline{p} \in \overline{\mathbb{P}}$ of type i), starting at the source. Denoting by v the last vertex on partial resource path $\overline{p}$, the reduced costs are

$$c_{\overline{p}} - \sum_{u \in V'} \lambda_u a_{u\overline{p}} - \mu_v - \pi_v \overline{y}^v_{\overline{p}1}.$$

We can generate this type of partial resource path with a minimal reduced cost as follows. Consider every pair of potential starting task u and ending task v, for $u, v \in V'$, such that the duty length satisfies $t^E_v - t^S_u \leq L$. Any path in $G' = (V, A \backslash A^B)$ starting with arc (s, u) and ending at vertex v or with arc (v, t), satisfies the duty length constraint as well as the cumulative work duration constraint, since $L \leq W$. Moreover, it does not use a break. To each arc $a \in A$, with tail $v \in V$, assign cost $c_a - \lambda_v$. A minimal reduced partial resource path of this type can now be found by solving a shortest path problem without resources for every suitable pair of tasks. Since $\pi_v \geq 0$, it is optimal to choose $\overline{y}^v_{\overline{p}1}$ as large as possible, and thus equal to minus the cumulative work duration on the path. Note that because there are $\mathcal{O}(|V'|^2)$ candidate pairs, and a shortest path problem without resource constraints on an acyclic graph can be solved in polynomial time, we can find the minimal reduced cost partial resource path of this type in polynomial time.

By a similar construction, also for a partial resource path of type ii), starting with a break arc and ending at the depot, the minimal reduced cost partial resource path can be found by solving at most $\mathcal{O}(|V'|^2)$ shortest path problems without resource constraints on an acyclic graph. We conclude that the pricing problem of (24)-(28) is polynomially solvable. □

7 A vehicle routing example

In this section, we provide another example of a partial path relaxation. We consider the CVRP, for which a partial path relaxation is provided in [17]. Here, we consider a very similar partial path reformulation and refer to it as a p-step formulation. The partial path formulation uses partial paths of roughly length p. In this example we numerically compare the LP bounds and computation time of the p-step formulation, with another partial path relaxation of roughly resource consumption q. Although the pricing problem of the latter is pseudo-polynomially solvable instead of polynomially, like the pricing problem of the p-step formulation, there might still be computational advantages.

7.1 Problem description

Consider a simple graph $G = (V, A)$ such that $V = V' \cup \{s, t\}$ and A consists of all arc (s, v), (u, v) and (v, t) for all $u, v \in V'$. The set V' represents a set of customers and s and t are the starting and ending depot respectively. Associated with each customer $v \in V'$ is a demand $0 \leq q_v$, and we define $q_s = q_t = 0$. An unlimited number of vehicles is available for satisfying demand. A vehicle traverses a path from s to t, referred to as a route, and satisfies demand of all customers on the route. The cumulative demand of all customers on the route may not exceed the vehicle capacity Q, and we assume that $q_v \leq Q$ for all $v \in V'$. We denote the set of all such routes as $\mathbb{P}$. With each arc $(u, v) \in A$ we associate a cost $c_{uv} \geq 0$, which is incurred when a vehicle traverses that arc. The capacitated vehicle routing problem (CVRP) is the problem of finding routes that together satisfy the demands of all customers at minimal total costs.

7.2 Path programming formulation

Next, we provide a path programming formulation for the CVRP. Let c_p denote the cost of a route $p \in \mathbb{P}$, and let a_{vp} indicate whether customer $v \in V'$ is included on route p. Furthermore, the binary variable x_p indicates whether route $p \in \mathbb{P}$ is selected. With these parameter and variable definitions, formulation (21)-(23) is a path programming formulation of the CVRP. This formulation is a common set partitioning formulation of the CVRP, of which the pricing problem is modeled as the following SPPRC.

Consider the graph G defined in Section 7.1. We introduce the single resource of non-negative load, by defining $S = \{0\}$ and $F^{(u,v)}(y) = \{y + q_v\} \cap [0, Q]$ for all arcs $(u, v) \in A$. We associate a cost with every arc $a \in A$. Finding the least cost resource path in G is an SPPRC, which we denote by SPPRC(2).

Similar to before, we observe that a route corresponds to a resource path in G, denote by λ the dual multipliers corresponding to the set partitioning constraints (22), and let $\lambda_t = 0$. We model the pricing problem of (21)-(23) as an SPPRC(2), by associating with each arc $a \in A$ with head $v \in V$ the cost $c_a - \lambda_v$. It is considered well-known that SPPRC(2), also known as the elementary shortest path problem with a capacity constraint, is NP-hard.

7.3 Partial path relaxations

Recall from Section 5.3 that a partial resource path of roughly resource consumption q, is the following. It is a partial resource path such that i) the resource consumption between the first and the semi-last vertex is at most q, while the resource consumption between the first and the last vertex is strictly larger than q, or ii) it ends at the sink and has a total resource consumption of at most q. For the CVRP, we refer to a partial resource path of roughly resource consumption q as a q-step. Furthermore, we set $S^v = [0, Q]$ for all $v \in V'$. Similarly, a p-step is defined as a partial resource path of roughly length p.

Denote by $\overline{\mathbb{P}}$ the set of all q-steps, and by $\overline{y}^v_{\overline{p}1}$ the resource parameter corresponding to the single monotonic resource, load. With these definitions, formulation (24)-(28) is a partial path reformulation of the CVRP, which we refer to as the q-step formulation. Replacing $\overline{\mathbb{P}}$ in (24)-(28) by the set of all p-steps, we obtain a formulation equivalent to the p-step formulation of [17].

Next, we compare the p-step formulation and the q-step formulation. The computational complexity of the pricing problems and the LP bounds are of primary interest. Note that the pricing problem of the p-step formulation is polynomially solvable, see [17]. However, for the pricing problem of the q-step problem, no polynomial time algorithm is known. Still, next we provide numerical experiments in which stronger LP bounds are achieved with the q-step formulation in less computation time.

To compute the LP bounds of the p-step and q-step formulation, we make use of the column generation algorithm presented in [23]. More specifically, we use the same code that was used to perform the numerical experiments in [23] to solve previously unsolved instances of the CVRP. We make slight modifications to generate p-steps and q-steps instead of full paths. Similar to the algorithm presented in [17], we solve the pricing problem by solving an SPPRC using a labeling algorithm for each of the $\mathcal{O}(|V'|^2)$ pairs of start and end vertices. This experiment is performed on a single core of an Intel Xeon CPU W-2123 3.60 GHz with 64 GB RAM running Microsoft Windows 10.

Table 1 provides the LP bounds of the p-step formulation (LP bound p) and q-step formulation (LP bound q) for varying values of p and q, as well as the corresponding time in seconds to compute the LP bounds (Time(s) p and Time(s) q). We use well-known benchmark instances [2, 11]. For ease of exposition, we parametrize the values of p and q. In particular, per instance we determine an upper bound on p, the maximum number of customers that fit in one vehicle, and use as upper bound on q the vehicle capacity Q. Using a percentage α, we compute p as α times the upper bound, rounded, plus one, and compute q as α times the upper bound rounded. In the six rightmost columns of Table 1, the first row shows the used percentages.

First note that we do not display the results for 0% and 100% in Table 1. For these cases the LP bounds of the p-step formulation and q-step formulation are by definition equal. We also remark that the lowest computation times in our experiments are always observed for one of these cases. For 0% it could be expected that computation times are low, because partial resource paths consist of one arc. For 100% we conjecture that low computation times are observed for the following reasons. The used code was originally designed to construct full paths, and employs techniques that are known to reduce computation time in that case, see [23]. Another reason is that the pricing algorithms solve $\mathcal{O}(|V'|^2)$ SPPRCs. In practice, for 100%, there is only one relevant SPPRC that needs to be solved, namely that from source to sink, which reduces computation time. In [17], for the p-step formulation it is demonstrated that parallellization can be used to reduce the computation time of the column generation algorithm for intermediate cases but not for the case of 100%.

Table 1: This table provides lower bounds and computation time in seconds for benchmark instances using the p-step formulation and the q-step formulation. The six rightmost columns correspond to the different values of p and q, defined as a percentage of an upper bound, where the percentage is provided in the top row.

Instance	Statistic		20%	40%	60%	80%	90%	95%
A-n39-k6	LP bound	p	764.0	785.1	806.7	806.7	806.7	806.7
		q	759.1	764.6	780.7	796.4	801.8	806.5
	Time(s)	p	2.5	24.3	18.5	29.6	35.1	47.2
		q	8.8	20.0	2.0	6.6	8.0	9.5
A-n44-k6	LP bound	p	882.5	910.9	927.1	927.1	927.1	927.1
		q	861.4	875.2	897.4	912.0	919.5	926.9
	Time(s)	p	3.9	30.1	14.4	33.2	75.2	81.4
		q	12.3	22.8	3.8	8.3	9.4	11.3
A-n45-k6	LP bound	p	853.6	900.3	929.3	929.3	929.3	929.3
		q	846.0	843.2	868.1	899.4	913.1	929.0
	Time(s)	p	5.6	32.8	50.5	188.1	350.8	400.2
		q	14.7	43.5	3.3	10.5	14.4	12.6
A-n45-k7	LP bound	p	1073.6	1111.2	1124.7	1124.7	1124.7	1124.7
		q	1058.0	1071.4	1095.2	1108.8	1114.4	1124.3
	Time(s)	p	3.8	48.3	24.8	46.7	36.7	36.8
		q	13.2	27.9	3.1	8.7	8.9	10.2
A-n46-k7	LP bound	p	860.1	898.9	904.6	904.6	904.6	904.6
		q	850.4	856.1	872.1	885.3	900.7	904.2
	Time(s)	p	5.1	88.8	31.4	72.3	41.3	92.1
		q	14.9	35.7	5.2	6.7	7.3	9.1
A-n48-k7	LP bound	p	1005.5	1045.9	1051.6	1053.1	1053.1	1053.1
		q	992.7	1001.2	1022.2	1039.8	1047.9	1050.5
	Time(s)	p	4.4	32.8	61.4	169.4	230.8	200.4
		q	16.8	40.6	5.0	7.8	14.5	17.7
E-n23-k3	LP bound	p	541.4	545.4	550.6	559.2	565.2	565.2
		q	541.0	542.0	552.0	556.3	562.2	563.5
	Time(s)	p	4.7	10.6	11.2	20.7	17.7	23.1
		q	96.9	185.4	42.4	105.7	116.8	182.0
E-n30-k3	LP bound	p	457.6	467.0	480.5	480.9	480.9	480.9
		q	457.8	457.1	460.6	468.6	476.3	479.2
	Time(s)	p	1.4	44.4	61.2	308.6	880.0	810.6
		q	8.2	41.2	9.4	10.8	15.7	28.1
E-n33-k4	LP bound	p	802.1	812.2	819.5	820.9	820.9	820.9
		q	788.1	791.8	801.5	810.3	815.3	818.1
	Time(s)	p	4.2	26.0	50.2	80.4	165.1	186.3
		q	45.2	109.4	28.3	65.8	89.3	74.0
E-n51-k5	LP bound	p	503.6	511.8	516.6	517.1	517.1	517.1
		q	500.2	503.3	510.0	512.4	514.1	516.8
	Time(s)	p	11.1	220.8	51.8	256.9	199.4	576.1
		q	37.2	105.3	21.5	47.4	60.6	57.2

We mainly wish to point out the following observation from Table 1. The q-step formulation sometimes achieves stronger bounds in less computation time. This is for example the case for instance A-n39-k6, when comparing p at 20% and q at 60%. This is interesting since the pricing problem of the p-step formulation can be solved in polynomial time, while for the q-step formulation we do not know of a polynomial time algorithm. Also note that the LP bounds of the p-step formulation increase faster for low percentages α than the LP bounds of the q-step formulation. This is due to the fact that the strongest bounds are obtained when a solution to either formulation makes use of routes with a cumulative demand close to the capacity. For q-step formulations such routes are obtained for the case close to 100%, while for the p-step formulation they are obtained for the case of a lower percentage.

8 Conclusions

The partial path relaxation is a framework which has many applications. A careful design allows a reduction of the computational complexity of a pricing problem, or indeed of computing the LP bound, at the expense of a weaker LP bound. We have shown that if the pricing problem of the path programming formulation is an SPPRC, a generic partial path relaxation always exists for which the pricing problem can be solved in polynomial time. In the future, application specific path programming relaxations might be constructed to attain more computational gains.

Conflict of interest

The authors declare that they have no conflict of interest.

References

[1] E. Abbink, M. Fischetti, L. Kroon, G. Timmer, and M. Vromans. Reinventing crew scheduling at Netherlands Railways. *Interfaces*, 35(1):393–401, Oct 2005.

[2] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 949-M, Université Joseph Fourier, Grenoble, France, 1995.

[3] L. Bach, M. Gendreau, and S. Wøhlk. Freight railway operator timetabling and engine scheduling. *European Journal of Operational Research*, 241(2):309 – 319, 2015.

[4] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115:351–385, 2008.

[5] R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011.

[6] E. Bartolini, J.-F. Cordeau, and G. Laporte. Improved lower bounds and exact algorithm for the capacitated arc routing problem. *Mathematical Programming*, 137:409–452, 2013.

[7] M. R. Bussieck, T. Winter, and U. T. Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming*, 79:415–444, 1997.

[8] V. Cacchiani, A. Caprara, and P. Toth. Solving a real-world train-unit assignment problem. *Mathematical Programming*, 124:207–231, 2010.

[9] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P. L. Guida. Algorithms for railway crew management. *Mathematical Programming*, 79(1-3):125–141, Oct 1997.

[10] E. Choi and D.-W. Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7):2080–2095, 2007.

[11] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, 20:309–318, 1969.

[12] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, Dec 1981.

[13] C. Contardo and R. Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, 12:129–146, 2014.

[14] G. Desaulniers, F. Errico, S. Irnich, and M. Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405, 2016.

[15] M. Desrochers. *La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes.* PhD thesis, 1986.

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

[17] T. Dollevoet, P. Munari, and R. Spliet. A p-step formulation of the capacitated vehicle routing problem. Technical Report EI-2020-01, Econometric Institute, 2019.

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

[19] R. Fukasawa, H. Longo, J. Lysgaard, M. P. d. Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, May 2006.

[20] I. Ioachim, S. Glinas, F. Soumis, and J. Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3):193–204, 1998.

[21] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, New York, 2005.

[22] S. Irnich and D. Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18(3):391–406, 2006.

[23] D. Pecin, A. Pessoa, M. Poggi, and E. Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100, Mar 2017.

[24] D. Potthoff, D. Huisman, and G. Desaulniers. Column generation with dynamic duty selection for railway crew rescheduling. *Transportation Science*, 44(4):493–505, 2010.

[25] D. M. Ryan and B. A. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, chapter 17, pages 269–280. North Holland, 1981.

[26] R. Spliet and A. F. Gabor. The time window assignment vehicle routing problem. *Transportation Science*, 49(4):721–731, 2015.