Robust two-stage combinatorial optimization problems under discrete demand uncertainties and consistent selection constraints

Christina Büsing, Sabrina Schmitz*

Combinatorial Optimization, RWTH Aachen University, Germany

Abstract

In this paper, we study a robust two-stage concept of combinatorial optimization problems under discrete demand uncertainty. Combinatorial optimization problems are based on a finite set of elements for which we decide whether they are part of a solution. We divide the elements into two types, the so-called fixed and free elements. In a first stage, we irrecoverably decide whether some fixed elements are part of a solution despite uncertain demand. In a second stage, we decide whether some free elements complete a solution after a demand scenario is realized. The objective is to find a robust solution that minimizes the worst-case cost over a finite scenario set.

We show the $\mathcal{NP}$-hardness of this robust two-stage version of several specific combinatorial optimization problems under discrete demand uncertainty but provide a polynomial-time solvable special case. In particular, we apply this concept to three combinatorial optimization problems: the representative multi-selection, the shortest path, and the minimum weight perfect b-matching problem. We prove the $\mathcal{NP}$-hardness and present special cases solvable in pseudo-polynomial and polynomial time.

Keywords: robust two-stage, discrete demand uncertainty, robust representative (multi-)selection problem, robust shortest path problem, robust minimum weight perfect b-matching problem

1. Introduction

Robust optimization facilitates (cost-efficient) decision-making for high-risk problems or problems with predefined goals where some parameters are subject to uncertainty. The uncertainty in the parameters is considered by a set of scenarios, each scenario represents plausible values that the parameters might realize. In conservative robust approaches, all decisions are made irrevocably despite uncertain parameters. The decisions are considered as robust if they are feasible for each scenario of the scenario set. In contrast, robust two-stage approaches distinguishes between decisions that must be made irrevocably despite uncertain parameters in a first stage and those that can be made after the uncertain parameters are realized in a second stage. First-stage decisions are considered as robust if they can be combined with second-stage decisions into a feasible solution regardless which scenario is realized. Second-stage decisions are made so that a feasible solution results for the realized scenario. The objective in robust optimization and robust two-stage optimization is to find a solution that is robust against uncertainties and minimizes the worst-case cost over the scenario set.

In this paper, we consider two-stage robustness for combinatorial optimization problems under discrete demand uncertainty. In combinatorial optimization problems, we decide for elements of a finite set whether they are part of a solution. Classical combinatorial optimization problems are, for example, the representative (multi-)selection problem, the shortest path problem, and the minimum weight perfect b-matching problem. In these problems,

*Corresponding author

Email address: `schmitz@combi.rwth-aachen.de` (Sabrina Schmitz)

a solution must satisfy so-called demand constraints. However, for combinatorial optimization problems whose demand is subject to uncertainty, we cannot determine a conservative robust solution, i.e., a solution that is feasible for all scenarios. For instance, in the shortest path problem, a minimum cost path is sought that connects a start vertex with an end vertex. Accordingly, we need to decide which arcs are part of a path. If the start and end vertices of the required path are uncertain, we cannot determine a path that is feasible for all realizations of the start and end vertices. For this reason, we introduce a robust approach to construct solutions in two stages to combinatorial optimization problems under discrete demand uncertainty. For this purpose, we divide the elements of a finite set into two types, the so-called fixed and free elements. In a first stage, we decide whether some fixed elements are part of a solution before a demand scenario is realized. The decisions about the fixed elements are, as the name implies, fixed and cannot be changed retroactively. In a second stage, we decide whether some free elements complete a solution after a demand scenario is realized. The decisions about the free elements are made so that, in combination with the first-stage decisions, they ensure a feasible and as best as possible solution for the realized demand scenario. We aim at a robust solution that minimizes the worst-case cost over a finite scenario set. Taking again the example of the shortest path problem, we divide the arcs into fixed and free arcs. For the fixed arcs, we must decide whether they are part of a path despite uncertain start and end vertices. These decisions must not be reversed. After the realization of the start and end vertices, we decide which free arcs complete the path.

The main contribution of this paper is summarized as follows. First, we present a general problem formulation for a robust two-stage concept of combinatorial optimization problems under discrete demand uncertainty. We refer to this concept as *robust two-stage combinatorial optimization under consistent selection constraints* (Rob2S≡). Second, we show that the Rob2S≡ version of several specific combinatorial optimization problems is $\mathcal{NP}$-hard but provide a polynomial-time solvable special case. Third, we consider the Rob2S≡ version of three combinatorial optimization problems: the representative multi-selection, the shortest path, and the minimum weight perfect b-matching problem. For the Rob2S≡ version of these three special cases, we analyze the complexity. We prove the $\mathcal{NP}$-hardness of these problems and present special cases solvable in (pseudo-)polynomial time.

The outline of this paper is as follows. In Section 2, we provide an overview of related work. In Section 3, we introduce notations and define the Rob2S≡ problem. In Section 4–6, we consider the Rob2S≡ version of the representative multi-selection, the shortest path, and the minimum weight perfect b-matching problem. In Section 7, we conclude our results.

2. Related work

In the literature, robust two-stage models are used to represent a variety of applications such as transportation [19], power system scheduling [7], and telecommunication problems [2, 35]. Adapted to specific applications, several versions of two-stage robustness are analyzed. For instance, Liebchen et al. [33] consider a recovery robust timetabling problem for railways with delay management in the second stage. Besides recoverable robustness, adjustable robustness, bulk robustness, and K-adaptability are established concepts for two-stage models in robust optimization [9].

In this paper, we consider the class of adjustable two-stage robustness, often referred to as just two-stage robustness. Ben-Tal et al. [3] introduce the extension from the single-stage robustness to the two-stage robustness where some of the decisions may be made (or may be adjusted) in a second stage after the uncertain parameters are realized. For an overview on adjustable robustness, we refer to the surveys of Buchheim and Kurtz [9], Kasperski and Zieliński [26], and İhsan Yanıkoğlu et al. [41]. Although problems modeled in two stages provide more flexibility compared to problems modeled in a single stage, the models are often difficult to solve. Even the robust two-stage versions of simple problems are $\mathcal{NP}$-hard [3]. For this reason, Ben-Tal et al. [6] formulate an approximation, the so-called affinely adjustable robust counterpart. They parameterize second-stage adjustable variables as affine functions of the uncertainty. Bertsimas and Caramanis [5, 16] introduce a framework of finite adaptability which can accommodate discrete variables. They parameterize the second-stage adjustable variables as piecewise constant functions of the uncertainty. Using examples, Bertsimas and Caramanis show that the affinely adjustable and finite adaptability frameworks are complementary. Exact solutions are obtained, for example, by Jiang et al. [21] and Thiele et al. [37] with variations of Benders' decomposition method. Furthermore, Zeng and Zhao [40] present a column-and-constraint generation algorithm which solves a sequence of mixed integer programs.

Unlike most of the robust (two-stage) approaches in the literature, we do not consider cost uncertainty but demand uncertainty. If demand is uncertain, the basic problem is to make first-stage decisions in such a way that second-stage decisions can still result in a feasible solution. Demand uncertainty is often studied in the context of network planning, for example, in telecommunications or road networks. A robust network is designed so that the capacity is large enough to route the demand of all scenarios. For instance, Atamtürk and Zhang [2] study a robust two-stage approach for network flow and design under demand uncertainty. They characterize robust decisions of the first stage with an exponential number of constraints and prove that the corresponding separation problem is $\mathcal{NP}$-hard, even for a network flow problem on a bipartite graph. For the special case that the second-stage network topology is totally ordered or an arborescence, they show that the separation problem is tractable. Other robust network design studies which determine the capacity under demand uncertainty are, for example, the studies of Álvarez-Miranda et al. [1], Cacchiani et al. [15], and Koster et al. [29]. In contrast, Büsing et al. [13] combine demand uncertainty with equal flow constraints for a robust two-stage extension of the minimum cost flow problem, the so-called robust minimum cost flow problem under consistent flow constraints (RobMCF≡). Before the demand is realized, a decision has to be made about how much flow is sent along some predetermined arcs, the so called fixed arcs, in the first stage. After the demand is realized, a decision is made about how much flow is sent along the remaining free arcs to obtain a feasible flow in the second stage. Unlike the deterministic minimum cost flow with minimum quantities problem [31, 36] and the robust network design studies mentioned above, the RobMCF≡ problem does not allow a minimum or maximum flow quantity on a fixed arc, respectively. Instead, the RobMCF≡ problem requires an equal flow value on a fixed arc. Büsing et al. [14] also analyze the uncapacitated version of the RobMCF≡ problem. For both the capacitated and uncapacitated versions, they prove the strong $\mathcal{NP}$-hardness on acyclic digraphs and the weak $\mathcal{NP}$-hardness on series-parallel digraphs. Some special cases are presented for which the problems are solvable in polynomial time. In this paper, we generalize the robust two-stage concept of Büsing et al. [14, 13] for combinatorial optimization problems under discrete demand uncertainty. Furthermore, we apply the concept to three combinatorial optimization problems: the representative (multi-)selection, the shortest path, and the minimum weight perfect b-matching problem. The robust single- and two-stage versions of these three problems are considered in the literature under cost uncertainty as shown by the following literature examples.

The robust representative selection problem is studied in a single-stage version, for example, by Dolgui and Kovalev [18], Deineko and Woeginger [17], and Kasperski et al. [25]. Goerigk et al. [20] analyze the robust two-stage version of the representative selection problem for convex uncertainty sets such as the polyhedral and ellipsoidal uncertainty set. They prove that the problem is strongly $\mathcal{NP}$-hard and two-approximable. Büsing [11] shows that the k-distance recoverable robust version is weakly $\mathcal{NP}$-hard for discrete scenario sets if the number of scenarios is constant. Furthermore, she proves that the exact subset recoverable robust version is solvable in polynomial time for Γ-scenarios.

The robust shortest path problem is studied, for example, by Yu and Yang [39]. They prove that the single-stage version of this problem is weakly $\mathcal{NP}$-hard for discrete scenarios, even if only two scenarios are considered. Karasan et al. [23] and Bertsimas and Sim [8] prove that the single-stage version of this problem is solvable in polynomial time for interval and Γ-scenarios, respectively. Büsing [12] proves that a recoverable robust version is strongly $\mathcal{NP}$-hard and not approximable. Goerigk et al. [20] prove the strong $\mathcal{NP}$-hardness of the robust two-stage version of the shortest path problem for the convex polyhedral and ellipsoidal uncertainty set.

For the robust matching problem, we refer to Kouvelis and Yu [30] who prove that the min-max assignment problem is $\mathcal{NP}$-hard even if only two scenarios are considered. Deineko and Woeginger [17] show that the min-max assignment problem with a fixed number of discrete scenarios is equivalent to the exact perfect matching problem. Kasperski et al. [24] prove that the rent-recoverable robust version is strongly $\mathcal{NP}$-hard, even if only two scenarios are considered. Katriel et al. [27] study two versions of a two-stage stochastic optimization matching problem on bipartite graphs where in the second stage either the edge cost or the vertices to be matched are uncertain.

3. Problem formulation

In this paper, we consider combinatorial optimization problems defined by a finite set U, a set of feasible solutions $\mathcal{F} \subseteq 2^U$, and a cost function $c : U \to \mathbb{R}$. In the following, we assume that the set of feasible solutions $\mathcal{F}$ is determined by an inequality system $\boldsymbol{Ax} \leq \boldsymbol{b}$ consisting of coefficient matrix $\boldsymbol{A} \in \mathbb{Z}^{m\times n}_{\geq 0}$ and demand vector $\boldsymbol{b} \in \mathbb{Z}^m$, labeled by $\mathcal{F}_{\boldsymbol{b}}$. We obtain the set of characteristic vectors corresponding to feasible solutions by $\mathcal{X}_{\boldsymbol{b}} := \{\boldsymbol{x} \in \{0,1\}^{|U|} : \boldsymbol{Ax} \leq \boldsymbol{b}\}$. The objective is to find a solution with minimum cost as shown by the following linear program

$$\min_{\boldsymbol{x}\in\mathcal{X}_{\boldsymbol{b}}} \boldsymbol{c}^T\boldsymbol{x}, \tag{1}$$

where $\boldsymbol{c} \in \mathbb{R}^{|U|}$ is the cost vector corresponding to cost function c.

In many practical applications of combinatorial optimization problems, decisions must be made even though the demand is subject to uncertainty. However, there are often decisions that can be postponed. Thus, problem (1) is often solved in multiple decision stages. In the following, we consider two decision stages for combinatorial optimization problems whose demand is subject to uncertainty. We represent the demand uncertainty by a finite set of scenarios $\mathcal{U} = \{\boldsymbol{b}^1, \dots, \boldsymbol{b}^k\} \subseteq \mathbb{Z}^m$. We assume that the finite set U, by which the combinatorial optimization problem is defined, is partitioned into sets U^{fix} and U^{free}, i.e., $U = U^{\text{fix}} \cup U^{\text{free}}$. We call the elements of the sets U^{fix} and U^{free} *fixed* and *free elements*, respectively. The two decision stages are defined as follows. In the first stage, we must decide for the fixed elements whether they are part of a solution before a demand scenario is realized. These decisions may not be reversed or changed. After the first decision stage, one of the scenarios of scenario set $\mathcal{U}$ is realized. In the second stage, we may then decide for the free elements whether they complete a solution. Therefore, the decision variables $\boldsymbol{x} \in \{0,1\}^{|U|}$ are divided into first-stage decision variables $\boldsymbol{x}^{\text{fix}} \in \{0,1\}^{|U^{\text{fix}}|}$ and second-stage decision variables $\boldsymbol{x}^{\text{free}} \in \{0,1\}^{|U^{\text{free}}|}$. For a first-stage decision $\tilde{\boldsymbol{x}}^{\text{fix}}$ and a realized demand $\tilde{\boldsymbol{b}}$, we define a set of feasible second-stage decisions that result in a feasible solution by the so-called *recourse action set* $\mathcal{R}_{\tilde{\boldsymbol{b}}}(\tilde{\boldsymbol{x}}^{\text{fix}}) = \{\boldsymbol{x}^{\text{free}} \in \{0,1\}^{|U^{\text{free}}|} \mid (\tilde{\boldsymbol{x}}^{\text{fix}}, \boldsymbol{x}^{\text{free}}) \in \mathcal{X}_{\tilde{\boldsymbol{b}}}\}$. Accordingly, a first-stage decision $\tilde{\boldsymbol{x}}^{\text{fix}}$ is a *feasible partial solution* to a realized demand $\tilde{\boldsymbol{b}}$ if the corresponding recourse action set $\mathcal{R}_{\tilde{\boldsymbol{b}}}(\tilde{\boldsymbol{x}}^{\text{fix}})$ is non-empty. For a realized demand $\tilde{\boldsymbol{b}}$, a set of feasible partial solutions is defined by $\mathcal{X}^{\text{fix}}_{\tilde{\boldsymbol{b}}} = \{\boldsymbol{x}^{\text{fix}} \in \{0,1\}^{|U^{\text{fix}}|} \mid \mathcal{R}_{\tilde{\boldsymbol{b}}}(\boldsymbol{x}^{\text{fix}}) \neq \emptyset\}$. We consider a first-stage decision $\boldsymbol{x}^{\text{fix}}$ as *robust* if it is a feasible partial solution for all demand vectors $\boldsymbol{b} \in \mathcal{U}$. Overall, we obtain a robust two-stage version of problem (1) whose objective is to minimize the cost of the worst-case scenario as follows

$$\min_{\boldsymbol{x}^{\text{fix}}\in \bigcap\limits_{\boldsymbol{b}\in\mathcal{U}} \mathcal{X}^{\text{fix}}_{\boldsymbol{b}}} \max_{\boldsymbol{b}\in\mathcal{U}} \min_{\boldsymbol{x}^{\text{free}}\in\mathcal{R}_{\boldsymbol{b}}(\boldsymbol{x}^{\text{fix}})} \boldsymbol{c}^T \begin{pmatrix} \boldsymbol{x}^{\text{fix}} \\ \boldsymbol{x}^{\text{free}} \end{pmatrix}. \tag{2}$$

We refer to problem (2) as *robust two-stage combinatorial optimization problem under consistent selection constraints* (Rob2S≡). We note that the second-stage decisions $\boldsymbol{x}^{\text{free}}$ vary depending on which scenario is realized. In contrast, the first-stage decisions $\boldsymbol{x}^{\text{fix}}$ stay the same. The Rob2S≡ problem implies that we need to determine feasible solutions $F_{\boldsymbol{b}} \in \mathcal{F}_{\boldsymbol{b}}$ for all demand scenarios $\boldsymbol{b} \in \mathcal{U}$ which satisfy the so-called *consistent selection constraints*

$$F_{\boldsymbol{b}} \cap U^{\text{fix}} = F_{\boldsymbol{b}'} \cap U^{\text{fix}} \text{ for all } \boldsymbol{b}, \boldsymbol{b}' \in \mathcal{U}.$$

For the ROB2S≡ version of some combinatorial optimization problems, even finding a feasible robust solution is difficult as stated in the following theorem. Proofs can be found in the following sections.

Theorem 3.1. *Finding a feasible solution to the* ROB2S≡ *version of the representative multi-selection, the shortest path, and the minimum weight perfect b-matching problem is strongly $\mathcal{NP}$-hard, even for an uncertainty set consisting of only two scenarios.*

Before concluding the section, we present a special case for which the ROB2S≡ problem is solvable in polynomial time.

Lemma 3.1. *Let* $\min\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in \{0,1\}^{|U|}\}$ *indicate a combinatorial optimization problem whose demand* $\boldsymbol{b}$ *is subject to uncertainty. The corresponding* ROB2S≡ *problem is solvable in polynomial time if matrix* $\boldsymbol{A}$ *is total unimodular and the number of fixed elements is constant.*

PROOF. For the set of discrete scenarios $\mathcal{U} = \{\boldsymbol{b}^1, \dots, \boldsymbol{b}^k\}$, the ROB2S≡ problem can be modeled as integer program as shown in the following. We use the notation $[k] := \{1, \dots, k\}$.

$$\begin{aligned}
\min \quad & C_{\max} & \\
\text{s.t.} \quad & \boldsymbol{c}^T \begin{pmatrix} \boldsymbol{x}^{\text{fix}} \\ \boldsymbol{x}_\lambda^{\text{free}} \end{pmatrix} \leq C_{\max} & \forall\, \lambda \in [k] \\
& \boldsymbol{A} \begin{pmatrix} \boldsymbol{x}^{\text{fix}} \\ \boldsymbol{x}_\lambda^{\text{free}} \end{pmatrix} \leq \boldsymbol{b}^\lambda & \forall\, \lambda \in [k] \\
& \boldsymbol{x}_\lambda^{\text{free}} \in \{0,1\}^{|U^{\text{free}}|} & \forall\, \lambda \in [k] \\
& C_{\max} \in \mathbb{R}_{\geq 0},\ \boldsymbol{x}^{\text{fix}} \in \{0,1\}^{|U^{\text{fix}}|} &
\end{aligned}$$

We relax the binary variables $\boldsymbol{x}_\lambda^{\text{free}}$ for all $\lambda \in [k]$ so that a mixed integer program (MIP) results. If we fix values of the the binary variables $\boldsymbol{x}^{\text{fix}}$, we can solve the resulting linear program in polynomial time and obtain an optimal solution as matrix $\boldsymbol{A}$ is total unimodular. Overall, we obtain an optimal solution in polynomial time by enumerating over the values of the constant number of the binary variables $\boldsymbol{x}^{\text{fix}}$. Alternatively, we can use the algorithm of Lenstra [32] to solve the MIP with a constant number of binary variables in polynomial time.

We note that the algorithm of Lenstra [32], suggested in the proof above, is also used to solve a MIP with a constant number of integer variables in polynomial time. Accordingly, we can use the algorithm for the ROB2S≡ version of discrete optimization problems whose variables are not binary but integer and whose matrix is total unimodular. We obtain a special case solvable in polynomial time for the ROB2S≡ version of special discrete optimization problems as shown in the following corollary.

Corollary 3.1. *The* ROB2S≡ *version of the minimum cost flow problem [28], the transshipment problem [28], and the minimum weight perfect b-matching problem on bipartite graphs [28] is solvable in polynomial time if the number of fixed elements is constant.*

4. Robust representative multi-selection problem under consistent selection constraints

In this section, we consider the ROB2S≡ version of the representative multi-selection problem under discrete demand uncertainty, which we call *robust representative multi-selection problem under consistent selection constraints* (ROBRS≡). In Section 4.1, we introduce notations and the problem definition. In Section 4.2, we analyze the complexity of the problem.

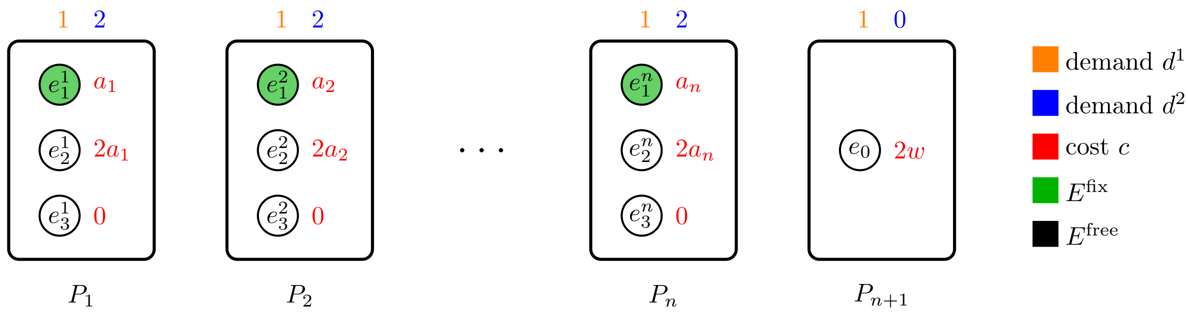

Figure 1: Construction of RobRS≡ instance $\widetilde{\mathcal{I}}$

4.1. Notations and problem definition

In the following, we consider the representative multi-selection problem defined as follows. Let a finite set of elements E be given. For each element of set E, let *cost* $c : E \to \mathbb{Z}_{\geq 0}$ be given. Furthermore, a *partition* P of the elements of set E into pairwise disjoint subsets is given, i.e., $P = \{P_1, \dots, P_r\}$ with $P_i \subseteq E$ for $i \in [r]$. For each subset of partition P, let *demand* $d : P \to \mathbb{Z}_{\geq 0}$ be given. A *representative multi-selection* (RM-selection) is defined by a subset of elements $F \subseteq E$ that satisfies the *demand constraints* $|F \cap P_i| = d(P_i)$ for every subset $P_i \in P$. The cost of a RM-selection F is defined by $c(F) = \sum_{e \in F} c(e)$. The objective is to find a RM-selection of minimum cost.

In the next step, we consider the Rob2S≡ version of the representative multi-selection problem. For this purpose, let a partition of set E into two disjoint sets E^{fix} and E^{free} be given. We call the elements of sets E^{fix} and E^{free} *fixed* and *free elements*, respectively. We assume that the demand is subject to uncertainty represented by the finite scenario set $\mathcal{U} = \{d^1, \dots, d^k\}$ of demands $d^\lambda : P \to \mathbb{Z}_{\geq 0}$ for all scenarios $\lambda \in [k]$. We use the notation $\boldsymbol{d} = (d^1, \dots, d^k)$. A *robust RM-selection* $\boldsymbol{F} = (F^1, \dots, F^k)$ is defined by a $|\mathcal{U}|$-tuple of RM-selections F^λ that satisfy the *consistent selection constraints* $F^\lambda \cap E^{\text{fix}} = F^{\lambda'} \cap E^{\text{fix}}$ for all scenarios $\lambda, \lambda' \in [k]$. The cost of a robust RM-selection $\boldsymbol{F}$ is defined by $c(\boldsymbol{F}) = \max_{\lambda \in [k]} c(F^\lambda)$. Finally, the RobRS≡ problem is formulated as follows.

Definition 4.1 (RobRS≡ problem). *Given an instance $(E = E^{fix} \cup E^{free}, c, P, \boldsymbol{d})$, the robust representative multi-selection problem under consistent selection constraints aims at finding a robust RM-selection of minimum cost.*

Before concluding this section we note the following. The representative (multi-)selection problem can be interpreted as a special case of several classical combinatorial optimization problems such as the perfect b-matching problem. This also holds true for the RobRS≡ problem as shown in Section 6.

4.2. Complexity

In this section, we analyze the complexity of the RobRS≡ problem. First, we prove that in general the decision version of the RobRS≡ problem is weakly $\mathcal{NP}$-hard as shown in the following theorem.

Theorem 4.1. *The* RobRS≡ *problem is weakly $\mathcal{NP}$-hard.*

Proof. The RobRS≡ problem is included in $\mathcal{NP}$ as we can check in polynomial time whether a set exists that satisfies the demand and consistent selection constraints. We perform a reduction from the weakly $\mathcal{NP}$-complete partition problem [22]. Let $\mathcal{I}$ be a partition instance defined by n positive integers $a_1, \dots, a_n$ that sum up to $\sum_{i=1}^n a_i = 2w$. Partition asks whether there exists a partition of set A into two disjoint subsets A_1 and A_2 such that the sum of the integers of each subset is equal, i.e., $w = \sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$. We construct a RobRS≡ instance $\widetilde{\mathcal{I}}$ as visualized in Figure 1. The instance is based on a set of elements E which contains one dummy element e_0 as well as three elements per integer a_i, $i \in [n]$, i.e., $E = \{e_0, e_1^1, e_2^1, e_3^1, e_1^2, e_2^2, e_3^2, \dots, e_1^n, e_2^n, e_3^n\}$. We identify the first elements e_1^i, $i \in [n]$ per integer as fixed elements contained in set E^{fix} and the dummy element e_0

as well as the second and third elements e_2^i, e_3^i, $i \in [n]$ per integer as free elements contained in set E^{free}. For all elements $e \in E$, we define the following cost

$$c(e) = \begin{cases} 2w & \text{if } e = e_0, \\ a_i & \text{if } e = e_1^i,\ i \in [n], \\ 2a_i & \text{if } e = e_2^i,\ i \in [n], \\ 0 & \text{if } e = e_3^i,\ i \in [n]. \end{cases}$$

We partition the set of elements E into $n+1$ pairwise disjoint subsets $P_1, \ldots, P_{n+1}$ as follows. We insert the three elements e_1^i, e_2^i, e_3^i, $i \in [n]$ per integer into subset P_i. Furthermore, we insert the dummy element e_0 into subset P_{n+1}. For all subsets P_i, $i \in [n+1]$, we define demand $\boldsymbol{d} = (d^1, d^2)$ by

$$d^1(P_i) = 1 \text{ for } i \in [n+1], \qquad d^2(P_i) = \begin{cases} 2 & \text{for } i \in [n], \\ 0 & \text{for } i = n+1. \end{cases}$$

Overall, we obtain a feasible RobRS≡ instance $\widetilde{\mathcal{I}} = (E = E^{\text{fix}} \cup E^{\text{free}}, c, P, \boldsymbol{b})$ constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance if and only if there exists a feasible robust RM-selection for instance $\widetilde{\mathcal{I}}$ with cost at most $\beta = 3w$.

Let A_1, A_2 be a feasible partition for instance $\mathcal{I}$. For the first scenario, we define the RM-selection $F^1 = \{e_0\} \cup \{e_1^i \mid a_i \in A_1\} \cup \{e_3^i \mid a_i \in A_2\}$. The cost is

$$c(F^1) = \sum_{e \in F^1} c(e) = c(e_0) + \sum_{i \in [n]\colon\, a_i \in A_1} c(e_1^i) + \sum_{i \in [n]\colon\, a_i \in A_2} c(e_3^i) = 2w + \sum_{a_i \in A_1} a_i + 0 = 3w.$$

For the second scenario, we define the RM-selection $F^2 = \{e_1^i \mid a_i \in A_1\} \cup \{e_2^i \mid a_i \in A_2\} \cup \{e_3^i \mid i \in [n]\}$. The cost is

$$c(F^2) = \sum_{e \in F^2} c(e) = \sum_{i \in [n]\colon\, a_i \in A_1} c(e_1^i) + \sum_{i \in [n]\colon\, a_i \in A_2} c(e_2^i) + \sum_{i=1}^{n} c(e_3^i) = \sum_{a_i \in A_1} a_i + \sum_{a_i \in A_2} 2a_i + 0 = 3w.$$

Consequently, we have constructed a robust RM-selection $\boldsymbol{F} = (F^1, F^2)$ with cost $c(\boldsymbol{F}) = 3w$.

Conversely, let $\boldsymbol{F} = (F^1, F^2)$ be a robust RM-selection with cost $c(\boldsymbol{F}) = \max\{c(F^1), c(F^2)\} \leq 3w$ for instance $\widetilde{\mathcal{I}}$. Let I_{fix} denote the set of indices that indicate which of the fixed elements $e_1^1, \ldots e_1^n$ are included in robust RM-selection $\boldsymbol{F} = (F^1, F^2)$, i.e., $I_{\text{fix}} = \{i \in [n] \mid e_1^i \in F^1 \cap F^2\}$. The RM-selection F^1 selects one element of each subset P_i for $i \in [n+1]$. We note that the only element of set P_{n+1} contributes $2w$ to the cost. A lower bound on the cost is given by

$$c(F^1) = \sum_{e \in F^1} c(e) \geq \sum_{i \in I_{\text{fix}}} c(e_1^i) + c(e_0) = \sum_{i \in I_{\text{fix}}} a_i + 2w.$$

As $c(F^1) \leq 3w$ holds, we obtain $\sum_{i \in I_{\text{fix}}} a_i \leq w$. The RM-selection F^2 selects two elements of each subset P_i, $i \in [n]$ and no element of subset P_{n+1}. A lower bound on the cost is given by

$$\begin{aligned} c(F^2) = \sum_{e \in F^2} c(e) &\geq \sum_{i \in I_{\text{fix}}} c(e_1^i) + \sum_{i \in [n] \setminus I_{\text{fix}}} c(e_2^i) + \sum_{i=1}^{n} c(e_3^i) = \sum_{i \in I_{\text{fix}}} a_i + \sum_{i \in [n] \setminus I_{\text{fix}}} 2a_i + 0 = 2\sum_{i=1}^{n} a_i - \sum_{i \in I_{\text{fix}}} a_i \\ &= 4w - \sum_{i \in I_{\text{fix}}} a_i. \end{aligned}$$

As $c(F^2) \leq 3w$ holds, we obtain $\sum_{i \in I_{\text{fix}}} a_i \geq w$. Overall, we obtain $\sum_{i \in I_{\text{fix}}} a_i = w$. We define the sets $A_1 = \{a_i \mid i \in I_{\text{fix}}\}$ and $A_2 = \{a_i \mid i \in [n] \setminus I_{\text{fix}}\}$ which form a feasible partition.

Before concluding this section, we consider the special case where at most one element is to be selected from each set. This special case is solvable in polynomial time as shown in the following theorem.

Theorem 4.2. *The* RobRS≡ *problem is solvable in polynomial time if only zero or one element is demanded from every subset in every scenario.*

Proof. Let $\mathcal{I} = (E, c, P, \boldsymbol{d})$ be a RobRS≡ instance with $d^\lambda : P \to \{0,1\}$, $\lambda \in [k]$. We note that a robust RM-selection $\boldsymbol{F}$ for instance $\mathcal{I}$ is optimal if and only if set $\boldsymbol{F} \cap P_i$ is optimal for RobRS≡ instance $\mathcal{I}_{P_i} = (E \cap P_i, c, P_i, \boldsymbol{d})$ for all $P_i \in P$. For this reason, we may consider the case $P = E$, i.e., $|P| = 1$. Demand $\boldsymbol{d} = (d^1, \dots, d^k)$ specifies if one or zero elements are selected from subset P in scenario $\lambda \in [k]$. Clearly, only the minimum cost free element and the minimum cost fixed element are relevant for the selection. If the cost of the minimum cost fixed element is equal or greater than the cost of the minimum cost free element, only the minimum cost free element is relevant. We obtain an optimal robust RM-selection by the following selection. If $d^{\lambda'}(P) = 0$ holds for at least one scenario $\lambda' \in [k]$, we may only select the minimum cost free element in all scenarios $\lambda \in [k]$ for which $d^\lambda(P) = 1$ holds. Otherwise, the consistent selection constraints could not be satisfied. If $d^\lambda(P) = 1$ holds for all scenarios $\lambda \in [k]$, we may select the minimum of the minimum cost free element and the minimum cost fixed element for all scenarios $\lambda \in [k]$.

5. Robust shortest path problem under consistent selection constraints

In this section, we consider the Rob2S≡ version of the shortest path problem under discrete demand uncertainty, which we call *robust shortest path problem under consistent selection constraints* (RobSP≡). First, we introduce notations and the problem definition. Second, we analyze the complexity of the problem.

5.1. Notations and problem definition

In the following, we consider the shortest path problem based on the definition of Korte and Vygen [28]. Let a *digraph* $G = (V, A)$ be given with vertex set V and arc set A. If not explicitly defined, we specify the sets of vertices and arcs of a digraph G by $V(G)$, $A(G)$, respectively. For each arc of set A, let conservative *cost* $c : A \to \mathbb{Z}_{\geq 0}$ be given. For a start and end vertex pair (s,t), an *(s,t)-path* in digraph G is defined by a subgraph $p = (\{s = v_0, v_1, \dots, v_n, v_{n+1} = t\}, \{a_1, \dots, a_{n+1}\}) \subseteq G$ where $a_i = (v_{i-1}, v_i)$, $i \in [n+1]$ with $v_i \neq v_j$ for $i < j$, $i, j \in \{0, \dots, n+1\}$. We note that a path according to this definition does not allow vertex repetition and hence edge repetition. Such a path is often referred to as *simple path*. The cost of an (s,t)- path p is defined by $c(p) = \sum_{a \in A(p)} c(a)$. The objective is to find a shortest (s,t)-path, i.e., an (s,t)-path of minimum cost.

In the next step, we consider the Rob2S≡ version of the shortest path problem. For this purpose, let a partition of arc set A into two disjoint sets A^{fix} and A^{free} be given. We call the arcs of sets A^{fix} and A^{free} *fixed* and *free arcs*, respectively. If not explicitly defined, we specify the sets of fixed arcs and free arcs of a digraph G by $A^{\text{fix}}(G)$ and $A^{\text{free}}(G)$, respectively. We assume that the start and end vertex pair is subject to uncertainty represented by the finite scenario set $\mathcal{U} = \{(s^1, t^1), \dots, (s^k, t^k)\}$ of vertex pairs (s^λ, t^λ) for all scenarios $\lambda \in [k]$. We use the notation $(\boldsymbol{s}, \boldsymbol{t}) = ((s^1, t^1), \dots, (s^k, t^k))$. A *robust $(\boldsymbol{s}, \boldsymbol{t})$-path* $\boldsymbol{p} = (p^1, \dots, p^k)$ is defined by a $|\mathcal{U}|$-tuple of (s^λ, t^λ)-paths that satisfy the *consistent selection constraints* $A(p^\lambda) \cap A^{\text{fix}} = A(p^{\lambda'}) \cap A^{\text{fix}}$ for all scenarios $\lambda, \lambda' \in [k]$. The cost of a robust $(\boldsymbol{s}, \boldsymbol{t})$-path $\boldsymbol{p}$ is defined by $c(\boldsymbol{p}) = \max_{\lambda \in [k]} c(p^\lambda)$. If the start (end) vertices of all scenario paths are defined by the same vertex, we say that the problem has a *unique start (end) vertex*. Finally, the RobSP≡ problem is formulated as follows.

Definition 5.1 (RobSP≡ problem). *Given an instance $(G = (V, A = A^{fix} \cup A^{free}), c, (\boldsymbol{s}, \boldsymbol{t}))$, the robust shortest path problem under consistent selection constraints aims at finding a shortest robust $(\boldsymbol{s}, \boldsymbol{t})$-path, i.e., a robust $(\boldsymbol{s}, \boldsymbol{t})$-path of minimum cost.*

Before concluding this section, we note that the exclusion of vertex repetition in the definition of a (scenario) path may influence whether there exists a feasible solution to the RobSP≡ problem as shown in Figure 2.

5.2. Complexity

In this section, we analyze the complexity of the RobSP≡ problem. We perform a reduction from the simple path with specified arcs problem (sPsA) defined as follows.

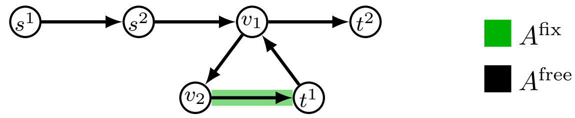

Figure 2: Example for an instance for which no feasible robust path exists because of the interdiction of vertex repetition

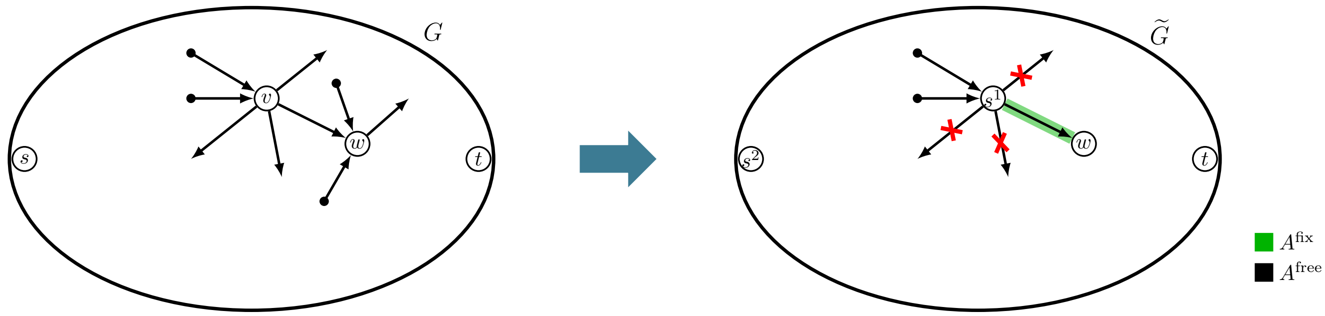

Figure 3: Transformation from sPsA instance $\mathcal{I}$ to RobSP≡ instance $\widetilde{\mathcal{I}}$.

Definition 5.2 (sPsA problem). *Let G be a digraph, $s, t \in V(G)$ be two specified vertices, and $a \in A(G)$ be a specified arc. The simple path with specified arcs problem asks whether there exists a simple (s,t)-path $p \subseteq G$ (without vertex repetition) which includes arc a, i.e., $a \in A(p)$.*

The sPsA problem is strongly $\mathcal{NP}$-complete which can be proven, for example, by a reduction from the strongly $\mathcal{NP}$-complete disjoint connecting path problem [22] as shown in Appendix A.

Theorem 5.1. *Deciding whether a feasible solution to the* RobSP≡ *problem exists is strongly $\mathcal{NP}$-complete, even if only one fixed arc and two scenarios with a unique end vertex are considered.*

Proof. The RobSP≡ problem is included in $\mathcal{NP}$ as we can check in polynomial time whether (s^λ, t^λ)-paths exist for all scenarios $\lambda \in [k]$ that satisfy the consistent selection constraints. We perform a reduction from the sPsA problem. Let $\mathcal{I}$ be an sPsA instance defined by a digraph G, two specified vertices $s, t \in V(G)$, and one specified arc $a^* = (v, w) \in A(G)$. We construct a RobSP≡ instance $\widetilde{\mathcal{I}}$ based on digraph $\widetilde{G}$ as visualized in Figure 3. We obtain digraph $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ from digraph G by deleting all outgoing arcs of vertex $v \in V(G)$ that are different from arc a^*, i.e., $\widetilde{G} = (\widetilde{V}, \widetilde{A})$ with $\widetilde{V} = V$ and $\widetilde{A} = A \setminus \{(v, u) \in A(G) \setminus \{a^*\}\}$. We specify arc a^* as the only fixed arc, i.e., $A^{\text{fix}}(\widetilde{G}) = \{a^*\}$ and $A^{\text{free}}(\widetilde{G}) = \widetilde{A} \setminus \{a^*\}$. We set the cost to $c \equiv 0$ and define the vertex pair $(\boldsymbol{s}, \boldsymbol{t}) = ((s^1, t^1), (s^2, t^2))$ by $s^1 = v$, $s^2 = s$, and $t^1 = t^2 = t$. Overall, we obtain a feasible RobSP≡ instance $\widetilde{\mathcal{I}} = (\widetilde{G}, c, (\boldsymbol{s}, \boldsymbol{t}))$ constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance for the sPsA problem if and only if there exists a feasible robust $(\boldsymbol{s}, \boldsymbol{t})$-path for instance $\widetilde{\mathcal{I}}$.

Let p be a feasible (s,t)-path for instance $\mathcal{I}$. We denote the subpath of path p from vertex v to vertex w by $p_{|vw}$. For the first scenario, we define path $p^1 = (\{v, w\}, \{a^*\}) \cup p_{|vw}$. For the second scenario, we set $p^2 = p$. Consequently, we have constructed a robust $(\boldsymbol{s}, \boldsymbol{t})$-path $\boldsymbol{p} = (p^1, p^2)$ for instance $\widetilde{\mathcal{I}}$.

Conversely, let $\boldsymbol{p} = (p^1, p^2)$ be a robust $(\boldsymbol{s}, \boldsymbol{t})$-path for instance $\widetilde{\mathcal{I}}$. Path p^1 is defined from vertex $s^1 = v$ to vertex $t^1 = t$. It includes necessarily the fixed arc a^* as it is the only outgoing arc of start vertex $s^1 = v$ in digraph $\widetilde{G}$. Due to the consistent selection constraints, path p^2 is defined from vertex $s^2 = s$ along arc a^* to vertex $t^2 = t$. Path p^2 is simple by definition of the scenario paths of a robust path. Consequently, path p^2 is a simple (s,t)-path in digraph G that contains arc $a^* \in A(G)$.

As a result of Theorem 5.1, we obtain that the problem is in general $\mathcal{NP}$-hard. In the next step, we consider the special case of the RobSP≡ problem where the underlying digraph is acyclic. In this special case, there exists an optimal robust path whose scenario paths either (1) include the same subpath which in turn includes all fixed arcs of the robust path or (2) include no fixed arcs, as shown in the following lemma.

Lemma 5.1. *Let a* RobSP≡ *instance $\mathcal{I} = (G, c, (\boldsymbol{s}, \boldsymbol{t}))$ be given, where $G = (V, A = A^{\text{fix}} \cup A^{\text{free}})$ is an acyclic digraph. There exists an optimal robust $(\boldsymbol{s}, \boldsymbol{t})$-path $\boldsymbol{p} = (p^1, \ldots, p^k)$ with one of the following properties*

(1) there exist two vertices $v, w \in V(G)$ such that $p^\lambda = \tilde{p}_{s^\lambda v} \cup p_{vw} \cup \tilde{p}_{wt^\lambda}$ with $p_{vw} \subseteq G$ and $\tilde{p}_{s^\lambda v}, \tilde{p}_{wt^\lambda} \subseteq G - A^{\text{fix}}$ holds for all scenarios $\lambda \in [k]$,

(2) $p^\lambda \subseteq G - A^{\text{fix}}$ holds for all scenarios $\lambda \in [k]$.

Proof. Let an optimal robust $(\boldsymbol{s}, \boldsymbol{t})$-path $\boldsymbol{p}$ be given. The robust path $\boldsymbol{p}$ includes fixed arcs or does not include fixed arcs. In the latter case, the robust path $\boldsymbol{p}$ satisfies the second property. In the first case, if necessary, we can transform robust $(\boldsymbol{s}, \boldsymbol{t})$-path $\boldsymbol{p} = (p^1, \dots, p^k)$ to a robust $(\boldsymbol{s}, \boldsymbol{t})$-path $\tilde{\boldsymbol{p}} = (\tilde{p}^1, \dots, \tilde{p}^k)$ that satisfies the first property as follows. The scenario paths $p^1, \dots, p^k$ include the same subset of fixed arcs, denoted by $\widetilde{A}^{\text{fix}} \subseteq A^{\text{fix}}$. In other words, the fixed arcs contained in subset $\widetilde{A}^{\text{fix}}$ are part of paths $p^1, \dots, p^k$. As the digraph is acyclic, the fixed arcs appear in the same order in every scenario path p^λ, $\lambda \in \Lambda$. Let index $i \in [r]$ indicate the order in which the fixed arcs $a_i^{\text{fix}} \in \widetilde{A}^{\text{fix}}$ with $|\widetilde{A}^{\text{fix}}| = r$ appear in the scenario paths. We denote the tail of fixed arc a_1^{fix} and the head of fixed arc a_r^{fix} by v and w, respectively. Let $p^\lambda_{|vw}$ denote the subpath of path p^λ from vertex v to vertex w. By the choice of vertices $v, w \in V(G)$, it holds $p^\lambda_{|s^\lambda v}, p^\lambda_{|wt^\lambda} \subseteq G - A^{\text{fix}}$ for all scenarios $\lambda \in [k]$. Assume there exist two scenarios $\lambda, \lambda' \in [k]$ in which the subpaths $p^\lambda_{|vw}$ and $p^{\lambda'}_{|vw}$ differ in at least one arc. Let p^*_{vw} denote the subpath with minimum cost among scenario paths $p^1_{|vw}, \dots, p^k_{|vw}$, i.e., $p^*_{vw} = \arg\min_{p \in \{p^1_{|vw}, \dots, p^k_{|vw}\}} c(p)$. A feasible robust $(\boldsymbol{s}, \boldsymbol{t})$-path is given by $\tilde{\boldsymbol{p}} = (\tilde{p}^1, \dots, \tilde{p}^k)$ with $\tilde{p}^\lambda = p^\lambda_{|s^\lambda v} \cup p^*_{vw} \cup p^\lambda_{|wt^\lambda}$, $\lambda \in [k]$, where $p^*_{vw} \subseteq G$ and $p^\lambda_{s^\lambda v}, p^\lambda_{wt^\lambda} \subseteq G - A^{\text{fix}}$ hold. By definition of subpath p^*_{vw}, it holds $c(\tilde{\boldsymbol{p}}) \leq c(\boldsymbol{p})$. Consequently, the transformation results in an optimal robust $(\boldsymbol{s}, \boldsymbol{t})$-path that satisfies the first property.

Using Lemma 5.1, we present the following algorithm to solve the RobSP≡ problem.

Algorithm 5.1

Input: RobSP≡ instance $(G, c, (\boldsymbol{s}, \boldsymbol{t}))$ where $G = (V, A = A^{\text{fix}} \cup A^{\text{free}})$ is an acyclic digraph

Output: Shortest robust $(\boldsymbol{s}, \boldsymbol{t})$-path

Initialization: Set $c_{\min} = \infty$

Method:

```
1: Compute shortest (v,w)-paths p_vw between all vertex pairs (v,w) ∈ V × V in digraph G
2: Compute shortest (v,w)-paths p̃_vw between all vertex pairs (v,w) ∈ V × V in digraph G − A^fix
3: for vertex pair (v,w) ∈ V × V do
4:     for scenario λ ∈ [k] do
5:         Compute cost c_{p^λ} = c(p̃_{s^λ v}) + c(p_vw) + c(p̃_{w t^λ})
6:     Compute c_max = max_{λ∈[k]} c_{p^λ}
7:     if c_max < c_min then
8:         Set c_min = c_max and p^λ = p̃_{s^λ v} ∪ p_vw ∪ p̃_{w t^λ} for all λ ∈ [k]
9: return robust (s,t)-path p* = (p^1, ..., p^k)
```

Algorithm 5.1 computes an optimal solution to the RobSP≡ problem in polynomial time as shown in the following theorem.

Theorem 5.2. *Let $\mathcal{I} = (G, c, (\boldsymbol{s}, \boldsymbol{t}))$ be a RobSP≡ instance where $G = (V, A = A^{\text{fix}} \cup A^{\text{free}})$ is an acyclic digraph. Algorithm 5.1 computes an optimal robust $(\boldsymbol{s}, \boldsymbol{t})$-path in polynomial time.*

Proof. First, the algorithm computes shortest paths between all vertex pairs in both digraph G and digraph $G - A^{\text{fix}}$. Second, the algorithm computes (the cost of) shortest scenario paths by specifying two vertices $v, w \in V(G)$ and composing shortest subpaths $\tilde{p}_{s^\lambda v}$, p_{vw}, and $\tilde{p}_{wt^\lambda}$ for all scenarios $\lambda \in [k]$ where $\tilde{p}_{s^\lambda v}, \tilde{p}_{wt^\lambda} \subseteq G - A^{\text{fix}}$ and $p_{vw} \subseteq G$ hold. The scenario paths composed satisfy the consistent selection constraints as they consist of the same subpath p_{vw} which include the same fixed arcs. By Lemma 5.1, there exists an optimal robust $(\boldsymbol{s}, \boldsymbol{t})$-path which satisfies property (1) or (2). The algorithm compares the shortest robust $(\boldsymbol{s}, \boldsymbol{t})$-paths that may include fixed arcs (corresponding to property (1)) with the shortest robust $(\boldsymbol{s}, \boldsymbol{t})$-path that does not include fixed arcs (corresponding to property (2)) by enumerating over all vertex pairs. We note that for all scenarios $\lambda \in [k]$ the shortest scenario paths $\tilde{p}_{s^\lambda t^\lambda} \subseteq G - A^{\text{fix}}$, which do not include fixed arcs, are composed by a vertex pair that consists of the same vertex. Consequently, the algorithm computes an optimal robust $(\boldsymbol{s}, \boldsymbol{t})$-path.

Considering the runtime of the algorithm, we obtain the following. The computation of all shortest paths between all vertex pairs in digraphs G and $G - A^{\text{fix}}$ is done each in $\mathcal{O}(|V|^3)$, for example, by the algorithm of Floyd and Warshall [28]. For each vertex pair, we compute the cost of all scenario paths and determine the maximum in $\mathcal{O}(k)$ time. We note that the number of scenarios is limited by the number of vertex pairs, i.e., $k \leq |V|^2$. In total, the computation of an optimal robust path takes $\mathcal{O}(|V|^4)$ time.

6. Robust minimum weight perfect $\boldsymbol{b}$-matching problem under consistent selection constraints

In this section, we consider the ROB2S≡ version of the minimum weight perfect b-matching problem, which we call *robust minimum weight perfect* $\boldsymbol{b}$*-matching problem under consistent selection constraints* (ROBPM≡). In Section 6.1, we introduce notations and the problem definition. In Section 6.2, we analyze the complexity of the problem.

6.1. Notations and problem definition

In the following, we consider the minimum weight perfect b-matching problem based on the definition of Korte and Vygen [28]. Let a *graph* $G = (V, E)$ be given with vertex set V and edge set E. For each vertex of set V, let *degree restriction* $b : V \to \mathbb{Z}_{\geq 0}$ be given. For each edge of set E, let *capacity* $u : E \to \mathbb{Z}_{\geq 0}$ and *weight* $c : E \to \mathbb{Z}_{\geq 0}$ be given. A *b-matching* in graph G is defined by a function $f : E \to \mathbb{Z}_{\geq 0}$ that satisfies the *capacity constraints* $0 \leq f(e) \leq u(e)$ for all edges $e \in E$ and the *degree constraints* $\sum_{e=\{v,w\}\in E} f(e) \leq b(v)$ for all vertices $v \in V$. A b-matching f is called *perfect* if the degree constraints are satisfied with equality for all vertices, i.e., $\sum_{e=\{v,w\}\in E} f(e) = b(v)$ for all $v \in V$. The weight of a b-matching f is defined by $c(f) = \sum_{e\in E} c(e)f(e)$. The objective is to find a perfect b-matching of minimum weight.

In the next step, we consider the ROB2S≡ version of the perfect b-matching problem. For this purpose, let a partition of the edge set E into two disjoint sets E^{fix} and E^{free} be given. We call the edges of sets E^{fix} and E^{free} *fixed* and *free edges*, respectively. If not explicitly defined, we specify the sets of vertices, edges, fixed edges, and free edges of a graph G by $V(G)$, $E(G)$, $E^{\text{fix}}(G)$, and $E^{\text{free}}(G)$, respectively. We assume that the degree restriction is subject to uncertainty represented by the finite scenario set $\mathcal{U} = \{b^1, \ldots, b^k\}$ of degree restrictions $b^\lambda : V \to \mathbb{Z}$ for all scenarios $\lambda \in [k]$. We use the notation $\boldsymbol{b} = (b^1, \ldots, b^k)$. A *robust perfect* $\boldsymbol{b}$*-matching* $\boldsymbol{f} = (f^1, \ldots, f^k)$ is defined by a $|\mathcal{U}|$-tuple of perfect b^λ-matchings $f^\lambda : E \to \mathbb{Z}_{\geq 0}$ that satisfy the *consistent selection constraints* $f^\lambda(e) = f^{\lambda'}(e)$ on all fixed edges $e \in E^{\text{fix}}$ for all scenarios $\lambda, \lambda' \in [k]$. The weight of a robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f}$ is defined by $c(\boldsymbol{f}) = \max_{\lambda\in[k]} c(f^\lambda)$. The ROBPM≡ problem is formulated as follows.

Definition 6.1 (ROBPM≡ problem). *Given an instance* $(G = (V, E = E^{fix} \cup E^{free}), u, c, \boldsymbol{b})$*, the robust minimum weight perfect* $\boldsymbol{b}$*-matching problem under consistent selection constraints aims at finding a robust perfect* $\boldsymbol{b}$*-matching of minimum weight.*

6.2. Complexity

In the paragraphs listed below, we analyze the complexity of the ROBPM≡ problem on bipartite, series-parallel (SP), pearl, and cactus graphs.

Complexity for bipartite graphs. In this section, we analyze the complexity of the ROBPM≡ problem on bipartite graphs. We perform a reduction from the strongly $\mathcal{NP}$-complete $(3, B2)$-Sat problem introduced by Berman et al. [4]. The $(3, B2)$-Sat problem is a special case of the 3-Sat problem [22] where every literal occurs exactly twice.

Theorem 6.1. *Deciding whether a feasible solution to the* ROBPM≡ *problem exists is strongly* $\mathcal{NP}$*-complete, even if only two scenarios are considered on a bipartite graph with zero-one degree restrictions.*

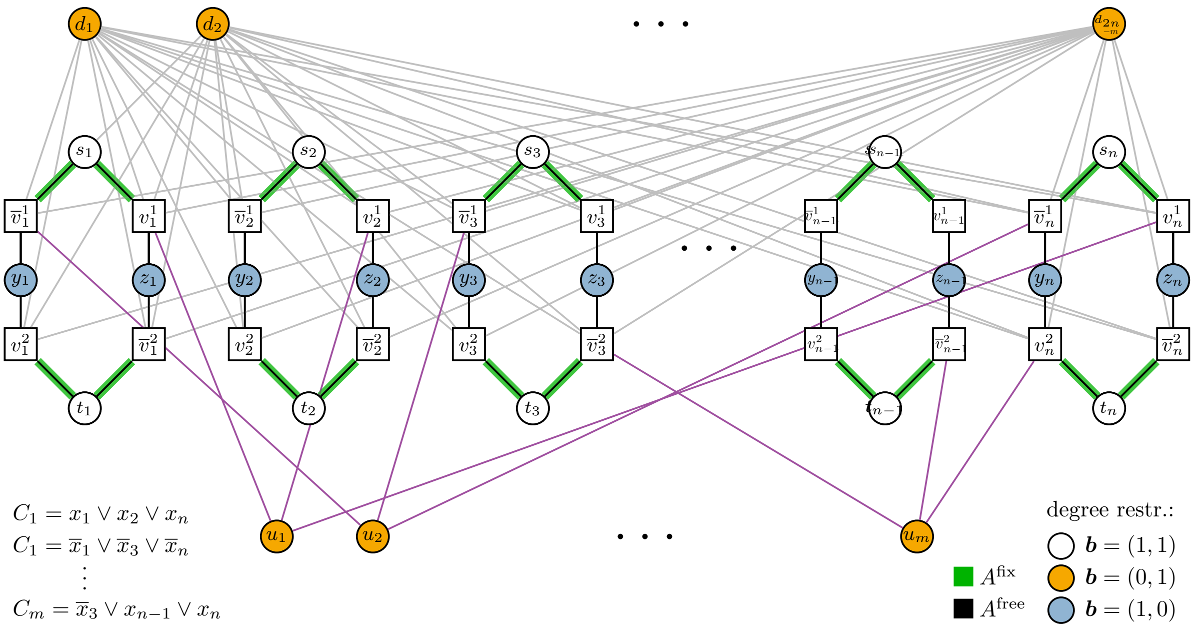

Figure 4: Construction of RobPM≡ instance $\widetilde{\mathcal{I}}$

Proof. The RobPM≡ problem is included in $\mathcal{NP}$ as we can check in polynomial time if the capacity constraints, the degree constraints with equality, and the consistent selection constraints are satisfied for all scenarios. Let $\{x_1, \dots, x_n\}$ be the set of variables and $C_1, \dots, C_m$ be the collection of clauses of a $(3, B2)$-Sat instance $\mathcal{I}$. $(3, B2)$-Sat asks whether there exists a satisfying truth assignment for all clauses. We construct a RobPM≡ instance $\widetilde{\mathcal{I}}$ as visualized in Figure 4. The instance is based on a graph $G = (V, E)$ defined as follows. Vertex set V includes four vertices s_i, t_i, y_i, z_i per variable x_i, two vertices v_i^1, v_i^2 per positive literal x_i, and two vertices $\overline{v}_i^1, \overline{v}_i^2$ per negative literal $\overline{x}_i$, $i \in [n]$. Furthermore, vertex set V includes one vertex u_j per clause C_j, $j \in [m]$ and $2n - m$ dummy vertices d_k, $k \in [2n - m]$. Edge set E contains three types of edges. The first type of edges connects the eight vertices which corresponds to variable x_i, $i \in [n]$ and create the cycle $X_i := s_i \overline{v}_i^1 y_i v_i^2 t_i \overline{v}_i^2 z_i v_i^1 s_i$. The second type of edges connects each dummy vertex d_k, $k \in [2n - m]$ with each of the literal vertices $\overline{v}_i^1, \overline{v}_i^2, v_i^1, v_i^2$, $i \in [n]$. The third type of edges connects each clause vertex u_j, $j \in [m]$ with the three vertices that represent the literals contained in clause C_j. The set of fixed edges E^{fix} is defined by the edges that are incident with vertices s_i and t_i for all $i \in [n]$, i.e., $E^{\text{fix}} = \{\{s_i, \overline{v}_i^1\}, \{s_i, v_i^1\}\{t_i, \overline{v}_i^2\}, \{t_i, v_i^2\} \mid i \in [n]\}$. The remaining edges are contained in the set of free edges E^{free}. We note that graph $G = (V, E)$ is bipartite with V partitioned into sets $U = \{s_i, y_i, z_i, t_i \mid i \in [n]\} \cup \{u_j \mid j \in [m]\} \cup \{d_k \mid k \in [2n - m]\}$ and $W = \{\overline{v}_i^\ell, v_i^\ell \mid \ell \in [2], i \in [n]\}$. The partition U, W is indicated in Figure 4 by circular and rectangular vertices. We set the capacity $u \equiv 1$ and the weight $c \equiv 0$. Finally, the degree restrictions $\boldsymbol{b} = (b^1, b^2)$ are defined as follows

$$b^1(v) = \begin{cases} 1 & \text{if } v \in V(X_i),\ i \in [n], \\ 0 & \text{otherwise,} \end{cases} \qquad b^2(v) = \begin{cases} 1 & \text{if } v \in V(G) \setminus \{y_i, z_i \mid i \in [n]\}, \\ 0 & \text{otherwise.} \end{cases}$$

Overall, we obtain a feasible RobPM≡ instance $\widetilde{\mathcal{I}}$ constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance if and only if a feasible robust perfect $\boldsymbol{b}$-matching exists for instance $\widetilde{\mathcal{I}}$.

Let $x_1, \dots, x_n$ be a satisfying truth assignment for instance $\mathcal{I}$. We denote literals x_i and $\overline{x}_i$, $i \in [n]$ which occur the ℓ-th time, $\ell \in [2]$ in the formula by x_i^ℓ and $\overline{x}_i^\ell$. For the first scenario, we define matching f^1 on all edges $e \in E$ by

$$f^1(e) = \begin{cases} 1 & \text{for } e \in \big\{\{s_i, \overline{v}_i^1\}, \{y_i, v_i^2\}, \{t_i, \overline{v}_i^2\}, \{z_i, v_i^1\}\big\} \text{ if } x_i = \textsc{True}, \\ 1 & \text{for } e \in \big\{\{s_i, v_i^1\}, \{z_i, \overline{v}_i^2\}, \{t_i, v_i^2\}, \{y_i, \overline{v}_i^1\}\big\} \text{ if } x_i = \textsc{False}, \\ 0 & \text{otherwise.} \end{cases}$$

For the second scenario, we define matching f^2 on the first part of edges E by

$$f^2(e) = \begin{cases} 1 & \text{for } e \in \{\{s_i, \overline{v}_i^1\}, \{t_i, \overline{v}_i^2\}\} \text{ if } x_i = \text{TRUE}, \\ 1 & \text{for } e \in \{\{s_i, v_i^1\}, \{t_i, v_i^2\}\} \text{ if } x_i = \text{FALSE}, \\ 1 & \text{for } e = \{v_i^\ell, u_j\} \text{ if } x_i^\ell \in C_j \text{ is verifying}, \\ 1 & \text{for } e = \{\overline{v}_i^\ell, u_j\} \text{ if } \overline{x}_i^\ell \in C_j \text{ is verifying}. \end{cases}$$

There are $2n - m$ literal vertices $\overline{v}_i^\ell, v_i^\ell$ that are not yet covered by matching f^2. Consequently, we can arbitrarily match these vertices with the $2n - m$ dummy vertices d_k, $k \in [2n - m]$ which are also not yet covered by matching f^2. We have constructed a feasible robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f} = (f^1, f^2)$.

Conversely, let $\boldsymbol{f} = (f^1, f^2)$ be a feasible robust perfect $\boldsymbol{b}$-matching for instance $\widetilde{\mathcal{I}}$. In the first scenario, only the vertices of cycles $X_1, \ldots, X_n$ are matched as all remaining vertices have degree restrictions of zero. For each cycle X_i, $i \in [n]$, only two matchings are perfect, namely, $\tilde{f}_i$ and $\hat{f}_i$ defined as follows

$$\tilde{f}_i(e) = \begin{cases} 1 & \text{for } e \in \{\{s_i, \overline{v}_i^1\}, \{y_i, v_i^2\}, \{t_i, \overline{v}_i^2\}, \{z_i, v_i^1\}\}, \\ 0 & \text{otherwise}, \end{cases}$$

$$\hat{f}_i(e) = \begin{cases} 1 & \text{for } e \in \{\{s_i, v_i^1\}, \{z_i, \overline{v}_i^2\}, \{t_i, v_i^2\}, \{y_i, \overline{v}_i^1\}\}, \\ 0 & \text{otherwise}. \end{cases}$$

In the second scenario, all vertices are matched except vertices y_i, z_i for all $i \in [n]$. In particular, the four literal vertices v_i^1, v_i^2, $\overline{v}_i^1$, $\overline{v}_i^2$ on each cycle X_i, $i \in [n]$ are matched but not with vertices y_i and z_i. Two of these four literal vertices are matched with vertices s_i and t_i. The remaining two literal vertices are matched each with either a clause vertex u_j, $j \in [m]$ or one of the dummy vertices d_k, $k \in [2n - m]$. Conversely, every clause vertex u_j, $j \in [m]$ is matched with one of the vertices that represent the literals contained in clause C_j, i.e., with one literal vertex of one cycle X_i for one $i \in [n]$. If vertex u_j, $j \in [m]$ is matched with vertex $\overline{v}_i^\ell$ (v_i^ℓ) for $i \in [n]$ and $\ell \in [2]$, we know the following about some of the remaining vertices of cycle X_i. If $\ell = 1$, vertex s_i is matched with vertex v_i^1 ($\overline{v}_i^1$). If $\ell = 2$, vertex t_i is matched with vertex v_i^2 ($\overline{v}_i^2$). In both cases, matching $\hat{f}_i$ ($\tilde{f}_i$) is implied on cycle X_i in the first scenario due to the consistent selection constraints. Accordingly, in the second scenario, vertex t_i is matched with vertex v_i^2 ($\overline{v}_i^2$) if $\ell = 1$ and vertex s_i is matched with v_i^1 ($\overline{v}_i^1$) if $\ell = 2$.

Depending on matching f^1 on the cycles X_i, we set the variables of the $(3, B2)$-Sat instance as follows. If $f^1(e) = \hat{f}_i(e)$ holds for $e \in E(X_i)$, we set $x_i = \text{FALSE}$. If $f^1(e) = \tilde{f}_i(e)$ holds for $e \in E(X_i)$, we set $x_i = \text{TRUE}$. The clauses $C_1, \ldots, C_m$ are satisfied due to the following reason. Matching f^2 covers all vertices $u_1, \ldots, u_m$ corresponding to clauses $C_1, \ldots, C_m$. If vertex u_j, $j \in [m]$ is matched with vertex $\overline{v}_i^\ell$, $i \in [n]$, $\ell \in [2]$ in the second scenario, matching $\hat{f}_i$ covers the cycle X_i in the first scenario. Clause C_j is satisfied as $\overline{x}_i^\ell \in C_j$ and $x_i = \text{FALSE}$. If vertex u_j, $j \in [m]$ is matched with vertex v_i^ℓ, $i \in [n]$, $\ell \in [2]$ in the second scenario, matching $\tilde{f}_i$ covers the cycle X_i in the first scenario. Clause C_j is satisfied as $x_i^\ell \in C_j$ and $x_i = \text{TRUE}$. We note that a truth assignment is still feasible if two clause vertices u_j, $u_{j'}$, $j, j' \in [m]$ are both matched with literal vertices of the same cycle X_i, $i \in [n]$. Because of the degree restrictions, the clause vertices are not matched with the same literal vertex. Instead, the vertices u_j, $u_{j'}$ may be matched with either the negative literal vertices $\overline{v}_i^1$, $\overline{v}_i^2$ or positive literal vertices v_i^1, v_i^2. If this were not true, there would exist either two unmatched vertices on cycle X_i and or the consistent selection constraints were violated. Overall, $x_1, \ldots, x_n$ is a satisfying truth assignment for instance $\mathcal{I}$.

As a result, we obtain that the RobPM≡ problem is in general strongly $\mathcal{NP}$-complete. For this reason, we consider special cases of the RobPM≡ problem based on further graph classes in the following paragraphs.

Complexity for SP graphs. In this section, we analyze the complexity of the RobPM≡ problem on SP graphs. We define SP graphs based on the edge SP multi-graphs definition of Valdes et al. [38] as follows.

Definition 6.2 (SP graph). *An SP graph is recursively defined as follows.*

1. *An edge $\{o, q\}$ is an SP graph with origin o and target q.*
2. *Let G_1 with origin o_1 and target q_1 and G_2 with origin o_2 and target q_2 be SP graphs. The graph that is constructed by one of the following two compositions of SP graphs G_1 and G_2 is itself an SP graph.*

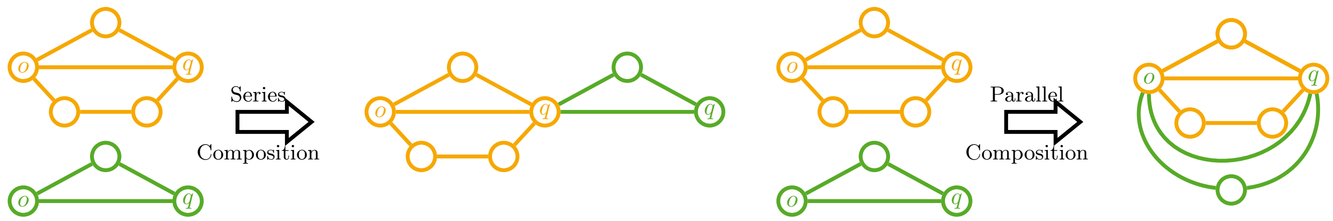

Figure 5: Example of an SP graph defined by a series or parallel composition

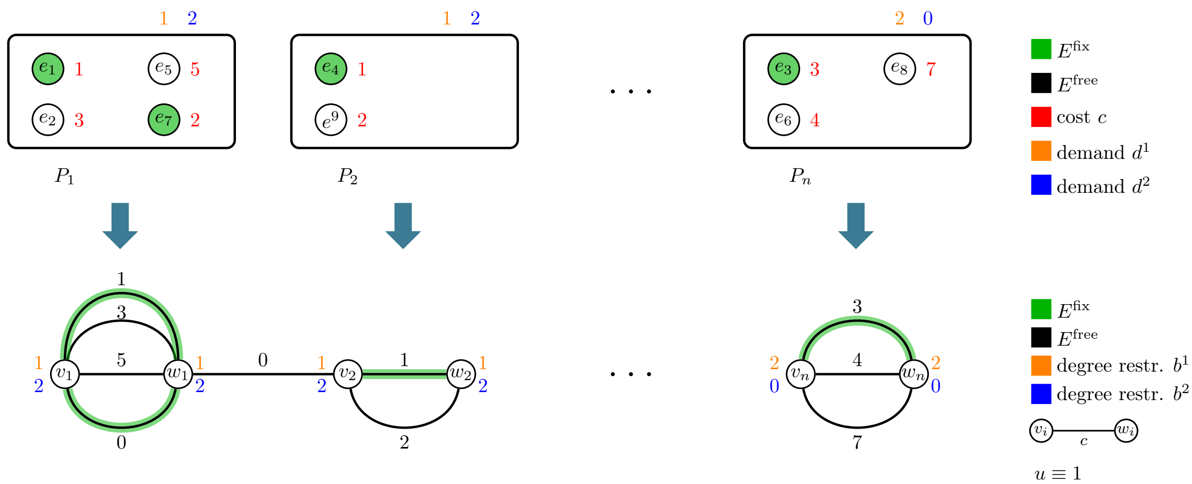

Figure 6: Transformation of a RobRS≡ instance $\mathcal{I}$ to a RobPM≡ instance $\widetilde{\mathcal{I}}$

a) *The series composition G of two SP graphs G_1 and G_2 is the graph obtained by contracting target q_1 and origin o_2. The origin of graph G is then o_1 (becoming o) and the target is q_2 (becoming q).*

b) *The parallel composition G of two SP graphs G_1 and G_2 is the graph obtained by contracting origins o_1 and o_2 (becoming o) and contracting targets q_1 and q_2 (becoming q). The origin of graph G is o and the target is q.*

The series and parallel compositions are illustrated in Figure 5. We note that in general SP graphs are multi-graphs with one definite origin and one definite target.

A complexity result for the RobPM≡ problem follows from the RobRS≡ problem which can be interpreted as a special case of the RobPM≡ problem on SP graphs as shown in the following corollary.

Corollary 6.1. *The* RobPM≡ *problem is weakly $\mathcal{NP}$-hard on SP graphs, even if only two scenarios are considered with unit edge capacities.*

Proof. Let $\mathcal{I} = (E = E^{\text{fix}} \cup E^{\text{free}}, c, P, \boldsymbol{d})$ be a RobRS≡ instance. The instance can be transformed to a specific RobPM≡ instance $\widetilde{\mathcal{I}} = (G, \tilde{c}, \boldsymbol{b})$ based on an SP graph $G = (V, \widetilde{E} = \widetilde{E}^{\text{fix}} \cup \widetilde{E}^{\text{free}})$ as visualized in Figure 6. SP graph G is a path of length $2|P| - 1$. For sake of simplicity, we identify the edges of the path in odd and even (multi-)edges. The ith edge with i even is a single dummy edge. The ith edge with i odd is a multi-edge that consists of $|P_i|$ parallel edges. We set the capacity of all edges to $u \equiv 1$. We construct a one-to-one correspondence between the free and fix elements of the RobRS≡ instance and the free and fix edges of the path's odd multi-edges of the RobPM≡ instance. Accordingly, we set the weight of the edges of an odd multi-edge to the corresponding element cost and the weight of the even edges to zero. Furthermore, we set the degree restrictions $\boldsymbol{b}$ for the incident vertices v_i, w_i of the $2i - 1$th edge to demand $\boldsymbol{d}$, i.e., $\boldsymbol{b}(v_i) = \boldsymbol{b}(w_i) = \boldsymbol{d}(P_i)$ for $i \in [|P|]$. Finally, it is easy to see that a Yes-Instance of the RobRS≡ problem is equivalent to a Yes-instance of the RobPM≡ problem.

We can formulate an even stronger complexity result on SP graphs as shown in the following theorem.

Theorem 6.2. *The* RobPM≡ *problem is weakly $\mathcal{NP}$-hard on SP graphs, even if only two scenarios are considered with zero-one degree restrictions.*

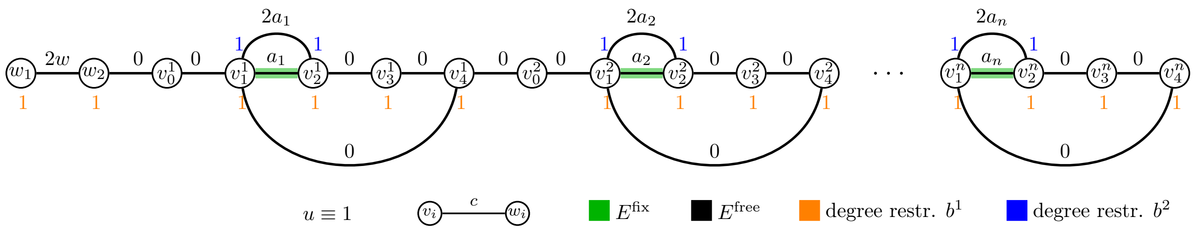

Figure 7: Construction of RobPM≡ instance $\widetilde{\mathcal{I}}$

Proof. The RobPM≡ problem is contained in $\mathcal{NP}$ as we can check in polynomial time if the capacity constraints, the degree constraints with equality, and the consistent selection constraints are satisfied for all scenarios. Let $\mathcal{I}$ be a partition instance defined by n positive integers $a_1, \ldots, a_n$ that sum up to $\sum_{i=1}^n a_i = 2w$. Partition asks whether there exists a partition of set A into two disjoint subsets A_1 and A_2 such that the sum of the integers of each subset is equal, i.e., $w = \sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$. We construct a RobPM≡ instance $\widetilde{\mathcal{I}}$ as visualized in Figure 7. The instance is based on an SP graph $G = (V, E)$ defined as follows. Vertex set V contains two auxiliary vertices w_1, w_2 and five vertices $v_0^i, v_1^i, v_2^i, v_3^i, v_4^i$ per integer a_i, $i \in [n]$. Edge set E includes edges that connect the vertices corresponding to integer a_i, $i \in [n]$ as follows. Two successive integer vertices v_{j-1}^i, v_j^i, $j \in [4]$ are connected by edge $e_j^i = \{v_{j-1}^i, v_j^i\}$. Vertices v_1^i,v_2^i are connected by an additional parallel edge $\tilde{e}_2^i = \{v_1^i, v_2^i\}$ and vertices v_1^i,v_4^i are connected by edge $\hat{e}^i = \{v_1^i, v_4^i\}$. Furthermore, edge set E includes the edges that connect vertices v_4^i, v_0^{i+1} for $i \in [n-1]$. Finally, edge set E includes two further edges that connect the two auxiliary vertices, i.e., $e^0 = \{w_1, w_2\}$, as well as vertices w_2 and v_0^1. The fixed edges of set E are defined by edges e_2^i for all $i \in [n]$ contained in set E^{fix}. The remaining edges are contained in set E^{free}. We set the capacity $u \equiv 1$ and define the weight c on all edges $e \in E$ by

$$c(e) = \begin{cases} 2w & \text{if } e = e^0, \\ 2a_i & \text{if } e = \tilde{e}_2^i,\ i \in [n], \\ a_i & \text{if } e = e_2^i,\ i \in [n], \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, we define the degree restrictions $\boldsymbol{b} = (b^1, b^2)$ on all vertices $v \in V$ by

$$b^1(v) = \begin{cases} 1 & \text{for all } v \in V \setminus \{v_0^i \mid i \in [n]\}, \\ 0 & \text{otherwise,} \end{cases} \qquad b^2(v) = \begin{cases} 1 & \text{if } v \in \{v_1^i, v_2^i \mid i \in [n]\}, \\ 0 & \text{otherwise.} \end{cases}$$

Overall, we obtain a feasible RobPM≡ instance $\widetilde{\mathcal{I}}$ constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance if and only if for instance $\widetilde{\mathcal{I}}$ a robust perfect $\boldsymbol{b}$-matching exists with weight of at most $\beta := 3w$.

Let A_1, A_2 be a feasible partition for instance $\mathcal{I}$. For the first scenario, we define matching f^1 by

$$f^1(e) = \begin{cases} 1 & \text{for } e = e^0, \\ 1 & \text{for } e = e_2^i \text{ and } e = e_4^i \text{ if } a_i \in A_1, \\ 1 & \text{for } e = e_3^i \text{ and } e = \hat{e}^i \text{ if } a_i \in A_2, \\ 0 & \text{otherwise.} \end{cases}$$

The weight is

$$\begin{aligned} c(f^1) &= \sum_{e \in E} c(e) f^1(e) \\ &= c(e^0) f^1(e^0) + \sum_{i \in [n]:\ a_i \in A_1} \big(c(e_2^i) f^1(e_2^i) + c(e_4^i) f^1(e_4^i)\big) + \sum_{i \in [n]:\ a_i \in A_2} \big(c(e_3^i) f^1(e_3^i) + c(\hat{e}^i) f^1(\hat{e}^i)\big) \\ &= 2w + \sum_{a_i \in A_1} (a_i + 0) + \sum_{a_i \in A_2} (0 + 0) = 3w. \end{aligned}$$

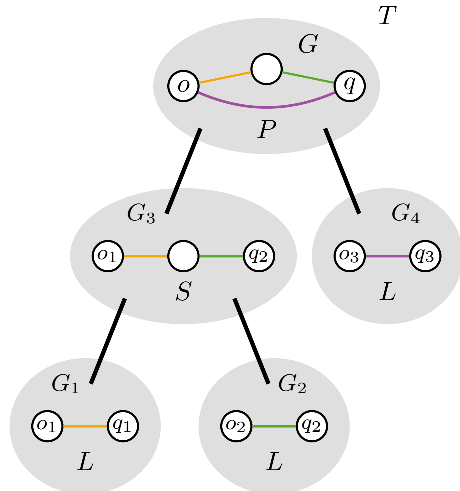

Figure 8: Example of the representation of an SP graph G by its SP tree T

For the second scenario, we define matching f^2 by

$$f^2(e) = \begin{cases} 1 & \text{for } e = e_2^i \text{ if } a_i \in A_1, \\ 1 & \text{for } e = \tilde{e}_2^i \text{ if } a_i \in A_2, \\ 0 & \text{otherwise.} \end{cases}$$

The weight is

$$c(f^2) = \sum_{e \in E} c(e) f^2(e) = \sum_{i \in [n]:\ a_i \in A_1} c(e_2^i) f^2(e_2^i) + \sum_{i \in [n]:\ a_i \in A_2} c(\tilde{e}_2^i) f^2(\tilde{e}_2^i) = \sum_{a_i \in A_1} a_i + \sum_{a_i \in A_2} 2a_i = 3w.$$

Consequently, we have constructed a robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f} = (f^1, f^2)$ with weight of $3w$.

Conversely, let $\boldsymbol{f} = (f^1, f^2)$ be a robust perfect $\boldsymbol{b}$-matching with weight $c(\boldsymbol{f}) = \max\{c(f^1), c(f^2)\} \le 3w$. For the first scenario matching f^1, it holds $f^1(e^0) = 1$ which contributes $2w$ to the weight. There are two possibilities to match the integer vertices. Either edges e_2^i and e_4^i or edges e_3^i and $\hat{e}^i$ are matched which causes a weight of a_i and 0, respectively. If edges e_3^i and $\hat{e}^i$ were covered for all $i \in [n]$ by matching f^1, matching f^2 might not contain a fixed edge e_2^i, $i \in [n]$ due to the consistent selection constraints. Instead, matching f^2 would contain edge $\tilde{e}_2^i$ for all $i \in [n]$ which causes a weight of

$$\sum_{i=1}^{n} c(\tilde{e}_i^2) = \sum_{i=1}^{n} 2a_i = 4w > 3w \ge \max\{c(f^1), c(f^2)\} = c(\boldsymbol{f}).$$

Thus, matching f^2 includes as many fixed edges e_2^i, $i \in [n]$ as weight of at least w is saved. In return, matching f^1 includes as many fixed edges e_i^2, $i \in [n]$ as weight of at most w is caused. As a result, the sets

$$A_1 = \{a_i \mid f^1(e_2^i) = 1 \text{ for } i \in [n]\},$$
$$A_2 = \{a_i \mid f^1(e_2^i) = 0 \text{ for } i \in [n]\}$$

form a feasible partition for instance $\mathcal{I}$.

In the next step, we refute the strong $\mathcal{NP}$-completeness of the RobPM≡ problem on SP graphs in the special case of a constant number of scenarios by presenting a pseudo-polynomial-time algorithm. For this purpose, we consider SP graphs represented by a rooted binary decomposition tree, a so-called *SP tree*, as shown in Figure 8. An SP tree indicates the composition of an SP graph by three different vertices, namely L-vertices, S-vertices, and P-vertices. Each edge of the SP graph is represented by an individual leaf of the SP tree, an L-vertex. The S- and P-vertices are the SP tree's inner vertices whose associated subgraphs are obtained by a series or parallel composition, respectively, of the subgraphs associated with their two child vertices. We note that the SP tree is constructed in polynomial time [38].

Using the SP tree, we propose an algorithm based on dynamic programming. The core idea of the dynamic program (DP) is a bottom-up method. The SP graph is composed step by step based on the SP tree. In each of these steps, a robust perfect matching is sought that satisfies three additional restrictions as explained in the following. First, the matching needs to satisfy the inner vertices' degree restrictions as their incident edges are already set. Second, the matching needs to satisfy new, smaller degree restrictions at the origin and target. We note that the original degree restrictions at the origin and target do not have to be satisfied yet as in subsequent steps further subgraphs may still be composed at these vertices. Third, the matching must exactly meet a weight budget given. The backtracking of the steps of the DP results in an optimal robust perfect $\boldsymbol{b}$-matching. Before presenting the DP in more detail, we introduce notations and labels.

Let $(G, u, c, \boldsymbol{b})$ be a RobPM≡ instance where G is an SP graph with origin o and target q. Let T be the SP tree of graph G. The subgraph of graph G associated to vertex $v \in V(T)$ is denoted by G_v. The origin and target of subgraph G_v are denoted by o_v and q_v, respectively. The DP relies on labels $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ defined for each subgraph G_v, $v \in V(T)$. The parameter vectors $\tilde{\boldsymbol{s}}_{\boldsymbol{v}} = (\tilde{s}_v^1, \ldots, \tilde{s}_v^k) \in \mathbb{Z}_{\geq 0}^k$ and $\tilde{\boldsymbol{t}}_{\boldsymbol{v}} = (\tilde{t}_v^1, \ldots, \tilde{t}_v^k) \in \mathbb{Z}_{\geq 0}^k$ determine new degree restrictions at origin o_v and target q_v of subgraph G_v, $v \in V(T)$, respectively. The new degree restrictions $\tilde{\boldsymbol{s}}_{\boldsymbol{v}}$ and $\tilde{\boldsymbol{t}}_{\boldsymbol{v}}$ may not be greater than degree restrictions $\boldsymbol{b}$, i.e., $\tilde{s}_v^\lambda \in \{0, \ldots, b^\lambda(o_v)\}$ and $\tilde{t}_v^\lambda \in \{0, \ldots, b^\lambda(q_v)\}$. The parameter vector $\tilde{\boldsymbol{c}}_{\boldsymbol{v}} = (\tilde{c}_v^1, \ldots, \tilde{c}_v^k) \in \mathbb{Z}_{\geq 0}^k$ specifies the weight budget that must be spent in subgraph G_v, $v \in V(T)$ with respect to weight c. An upper bound on the weight budget is given by the weight that can occur in subgraph G_v, i.e., $\tilde{c}_v^\lambda \in \{0, \ldots, C_v\}$ for $\lambda \in [k]$ with $C_v = B_v \sum_{e \in E(G_v)} c(e)$ where $B_v = \max_{\lambda \in [k], v \in V(G_v)} b^\lambda(v)$.

Using these parameters, we define the $(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$*-restricted* RobPM≡ *problem* (rRobPM≡ $(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$) as the RobPM≡ problem on subgraph G_v, $v \in V(T)$ with restrictions implied by degree restrictions $\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}$ and weight budget $\tilde{\boldsymbol{c}}_{\boldsymbol{v}}$. The label $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ is defined as the optimal solution value to the rRobPM≡ $(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem. For convenience and for the sake of clarity, we specify the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem by the following integer program. The integer program is formulated by means of integer variables $f_e^\lambda \in \mathbb{Z}_{\geq 0}$ which indicate the value of the matching on edge $e \in E(G_v)$ in scenario $\lambda \in [k]$.

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = \min \ 0 \tag{3}$$

$$\text{s.t.} \sum_{e \in E(G_v)} c(e) \cdot f_e^\lambda = \tilde{c}_v^\lambda \qquad \forall \lambda \in [k] \tag{4}$$

$$\sum_{e=\{w,z\} \in E(G_v)} f_e^\lambda = \begin{cases} b^\lambda(w) & \text{if } w \neq o_v,\ w \neq q_v, \\ \tilde{s}_w^\lambda & \text{if } w = o_v, \\ \tilde{t}_w^\lambda & \text{if } w = q_v, \end{cases} \qquad \forall w \in V(G_v), \lambda \in [k] \tag{5}$$

$$f_e^\lambda = f_e^{\lambda'} \qquad \forall e \in E^{\text{fix}}(G_v), \lambda, \lambda' \in [k] \tag{6}$$

$$0 \leq f_e^\lambda \leq u(e) \qquad \forall e \in E(G_v), \lambda \in [k] \tag{7}$$

$$f_e^\lambda \in \mathbb{Z}_{\geq 0} \qquad \forall e \in E(G_v), \lambda \in [k] \tag{8}$$

The rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem requires a robust perfect $\boldsymbol{b}$-matching in subgraph G_v by means of constraints (5)–(8). Therefore, the matching needs to satisfy the degree restrictions $\tilde{\boldsymbol{s}}_{\boldsymbol{v}}$ and $\tilde{\boldsymbol{t}}_{\boldsymbol{v}}$ at origin o_v and target q_v, respectively, and the degree restrictions $\boldsymbol{b}$ at all other vertices of subgraph G_v. Furthermore, the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem requires that the matching exactly meet the weight budget $\tilde{\boldsymbol{c}}_{\boldsymbol{v}}$ by means of constraints (4). The objective of the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem is to find a feasible solution, i.e., $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = 0$.

The DP uses the structure of the SP tree to compute labels recursively in a bottom-up procedure. For a specific vertex in the SP tree, the corresponding label is updated based on the labels corresponding to the child vertices. Depending on whether the SP tree's vertex considered is an L-, S-, or P-vertex, one of the following three procedures is applied. First, we consider the procedure in which the leaves are initialized.

Lemma 6.1. *Let $v \in V(T)$ be an L-vertex of SP tree T. The label $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ is initialized by*

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = \begin{cases} 0 & \text{if } \tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \tilde{\boldsymbol{t}}_{\boldsymbol{v}},\ \{o_v, q_v\} \in E^{\text{free}}(G_v), \text{ and } \tilde{c}_v^\lambda = c(\{o_v, q_v\}) \cdot \tilde{s}_v^\lambda,\ \forall \lambda \in [k], \\ 0 & \text{if } \tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \tilde{\boldsymbol{t}}_{\boldsymbol{v}},\ \{o_v, q_v\} \in E^{\text{fix}}(G_v),\ \tilde{s}_v^\lambda = \tilde{s}_v^{\lambda'}, \text{ and } \tilde{c}_v^\lambda = c(\{o_v, q_v\}) \cdot \tilde{s}_v^\lambda,\ \forall \lambda, \lambda' \in [k], \\ \infty & \text{otherwise.} \end{cases}$$

Proof. As $v \in V(T)$ is an L-vertex, subgraph G_v consists of the single edge $e_v := \{o_v, q_v\}$. First, if $\tilde{\boldsymbol{s}}_{\boldsymbol{v}} \neq \tilde{\boldsymbol{t}}_{\boldsymbol{v}}$ holds, there do not exist feasible matchings which satisfy constraints (5). In this case, the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem is not solvable, i.e., $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = \infty$.

If $e_v \in E^{\text{free}}(G_v)$, it must hold $\tilde{c}_v^\lambda = c(\{o_v, q_v\}) \cdot \tilde{s}_v^\lambda$ for all $\lambda \in [k]$. Otherwise, there do not exist feasible matchings that satisfy constraints (4) due to constraints (5). If $e_v \in E^{\text{fix}}(G_v)$, the constraints of the previous case need to be satisfied due to the same argumentation. In addition, $\tilde{s}_v^\lambda = \tilde{s}_v^{\lambda'}$ (and thus $\tilde{t}_v^\lambda = \tilde{t}_v^{\lambda'}$) must hold true for all scenarios $\lambda, \lambda' \in [k]$ due to constraints (6). If the presented constraints are satisfied for a fixed or a free edge $e_v \in E(G_v)$ a feasible optimal solution to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem is given by $\boldsymbol{f}(e_v) := \tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \tilde{\boldsymbol{t}}_{\boldsymbol{v}}$, i.e., $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = 0$.

Second, we consider the procedure in which a label is derived recursively from the labels of the child vertices that are composed in parallel.

Lemma 6.2. *Let $v \in V(T)$ be a P-vertex in SP tree T with child vertices $x, y \in V(T)$. The label $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ at vertex v can be computed by a composition of the labels $d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and $d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ of the child vertices x and y as follows*

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = \min_{\substack{\tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \tilde{\boldsymbol{s}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{v}} = \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{t}}_{\boldsymbol{y}} \\ \tilde{\boldsymbol{c}}_{\boldsymbol{v}} = \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}}} \left\{ d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \right\}.$$

Proof. For vertex $v \in V(T)$, let $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ be a label with the related solution $\boldsymbol{f}^*$. We partition matching $\boldsymbol{f}^*$ and the associated degree use $\tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \sum_{e=\{o_v,w\} \in E(G_v)} \boldsymbol{f}^*(a)$ at origin o_v and $\tilde{\boldsymbol{t}}_{\boldsymbol{v}} = \sum_{e=\{q_v,w\} \in E(G_v)} \boldsymbol{f}^*(a)$ at target q_v into two matchings $\boldsymbol{f}_{\boldsymbol{x}}$ and $\boldsymbol{f}_{\boldsymbol{y}}$ with associated degree use $\tilde{\boldsymbol{s}}_{\boldsymbol{x}}$, $\tilde{\boldsymbol{t}}_{\boldsymbol{x}}$ and $\tilde{\boldsymbol{s}}_{\boldsymbol{y}}$, $\tilde{\boldsymbol{t}}_{\boldsymbol{y}}$ in subgraphs G_x and G_y, respectively. The weight budget $\tilde{\boldsymbol{c}}_{\boldsymbol{v}} = \sum_{e \in E(G_v)} c(e) \cdot \boldsymbol{f}^*(e)$ of matching $\boldsymbol{f}^*$ is divided such that $\tilde{\boldsymbol{c}}_{\boldsymbol{x}}$ and $\tilde{\boldsymbol{c}}_{\boldsymbol{y}}$ describe the budgets of matchings $\boldsymbol{f}_{\boldsymbol{x}}$ and $\boldsymbol{f}_{\boldsymbol{y}}$, respectively. Matchings $\boldsymbol{f}_{\boldsymbol{x}}$ and $\boldsymbol{f}_{\boldsymbol{y}}$ are feasible solutions to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ problem, respectively. We obtain

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \geq d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}),$$

where $d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and $d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ are the labels corresponding to the child vertices $x, y \in V(T)$. In particular, this implies

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) \geq \min_{\substack{\tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \tilde{\boldsymbol{s}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{v}} = \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{t}}_{\boldsymbol{y}} \\ \tilde{\boldsymbol{c}}_{\boldsymbol{v}} = \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}}} \left\{ d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \right\}.$$

Conversely, for child vertices $x, y \in V(T)$, let $d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and $d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ be labels with related solutions $\boldsymbol{f}^*_{\boldsymbol{x}}$ and $\boldsymbol{f}^*_{\boldsymbol{y}}$. The combination of the matchings $\boldsymbol{f}^*_{\boldsymbol{x}}$ and $\boldsymbol{f}^*_{\boldsymbol{y}}$ results in a feasible solution $\boldsymbol{f}_{\boldsymbol{v}} := \boldsymbol{f}^*_{\boldsymbol{x}} + \boldsymbol{f}^*_{\boldsymbol{y}}$ to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem with degree use $\tilde{\boldsymbol{s}}_{\boldsymbol{v}} := \tilde{\boldsymbol{s}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}}$ at origin o_v and $\tilde{\boldsymbol{t}}_{\boldsymbol{v}} := \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{t}}_{\boldsymbol{y}}$ at target q_v and weight budget $\tilde{\boldsymbol{c}}_{\boldsymbol{v}} := \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}$. We obtain

$$d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \geq d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) = d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}),$$

where $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ is the label corresponding to vertex $v \in V(T)$. This implies

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) \le \min_{\substack{\tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \tilde{\boldsymbol{s}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{v}} = \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{t}}_{\boldsymbol{y}} \\ \tilde{\boldsymbol{c}}_{\boldsymbol{v}} = \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}}} \left\{ d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \right\}.$$

Third, we consider the procedure in which a label is derived recursively from the labels of the child vertices that are serially composed.

Lemma 6.3. *Let $v \in V(T)$ be an S-vertex in SP tree T with child vertices $x, y \in V(T)$. The label $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ at vertex v can be computed by a composition of the labels $d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and $d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ of the child vertices x and y as follows*

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = \min_{\substack{\tilde{\boldsymbol{s}}_{\boldsymbol{x}} = \tilde{\boldsymbol{s}}_{\boldsymbol{v}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{y}} = \tilde{\boldsymbol{t}}_{\boldsymbol{v}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} = \boldsymbol{b}(q_x) \\ \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}} = \tilde{\boldsymbol{c}}_{\boldsymbol{v}}}} \left\{ d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \right\}.$$

PROOF. Without loss of generality, we assume for vertices $v, x, y \in V(T)$ that graph G_v is constructed by contracting the target q_x of subgraph G_x with the origin o_y of subgraph G_y. For vertex $v \in V(T)$, let $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ be a label with the related solution $\boldsymbol{f}^*$. We partition matching $\boldsymbol{f}^*$ and the associated degree use $\tilde{\boldsymbol{s}}_{\boldsymbol{v}} = \sum_{e=\{o_v,w\} \in E(G_v)} \boldsymbol{f}^*(a)$ at origin o_v and $\tilde{\boldsymbol{t}}_{\boldsymbol{v}} = \sum_{e=\{q_v,w\} \in E(G_v)} \boldsymbol{f}^*(a)$ at target q_v into two matchings $\boldsymbol{f}_{\boldsymbol{x}}$ and $\boldsymbol{f}_{\boldsymbol{y}}$ where matching $\boldsymbol{f}_{\boldsymbol{x}}$ is defined on subgraph G_x and matching $\boldsymbol{f}_{\boldsymbol{y}}$ is defined on subgraph G_y. More precisely, we obtain matching $\boldsymbol{f}_{\boldsymbol{x}}(e) := \boldsymbol{f}^*(e)$ for all edge $e \in E(G_x)$ with associated degree use $\tilde{\boldsymbol{s}}_{\boldsymbol{x}} = \tilde{\boldsymbol{s}}_{\boldsymbol{v}}$, $\tilde{\boldsymbol{t}}_{\boldsymbol{x}} = \sum_{e=(q_x,z) \in E(G_x)} \boldsymbol{f}^*(e)$ in subgraph G_x. Furthermore, we obtain matching $\boldsymbol{f}_{\boldsymbol{y}}(e) := \boldsymbol{f}^*(e)$ for all edges $e \in E(G_y)$ with associated degree use $\tilde{\boldsymbol{s}}_{\boldsymbol{y}} = \sum_{e=\{o_y,z\} \in E(G_y)} \boldsymbol{f}^*(e)$, $\tilde{\boldsymbol{t}}_{\boldsymbol{y}} = \tilde{\boldsymbol{t}}_{\boldsymbol{v}}$ in subgraph G_y. We note that $\tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} = \boldsymbol{b}(q_x) = \boldsymbol{b}(o_y)$ holds. The weight budget $\tilde{\boldsymbol{c}}_{\boldsymbol{v}} = \sum_{e \in E(G_v)} c(e) \cdot \boldsymbol{f}^*(e)$ of matching $\boldsymbol{f}^*$ is divided such that $\tilde{\boldsymbol{c}}_{\boldsymbol{x}}$ and $\tilde{\boldsymbol{c}}_{\boldsymbol{y}}$ indicate the budgets of matchings $\boldsymbol{f}_{\boldsymbol{x}}$ and $\boldsymbol{f}_{\boldsymbol{y}}$, respectively. Matchings $\boldsymbol{f}_{\boldsymbol{x}}$ and $\boldsymbol{f}_{\boldsymbol{y}}$ are feasible solutions to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ problem, respectively. We obtain

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) = d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \ge d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}),$$

where $d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and $d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ are the labels corresponding to child vertices $x, y \in V(T)$. In particular, this implies

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) \ge \min_{\substack{\tilde{\boldsymbol{s}}_{\boldsymbol{x}} = \tilde{\boldsymbol{s}}_{\boldsymbol{v}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{y}} = \tilde{\boldsymbol{t}}_{\boldsymbol{v}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} = \boldsymbol{b}(q_x) \\ \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}} = \tilde{\boldsymbol{c}}_{\boldsymbol{v}}}} \left\{ d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \right\}.$$

Conversely, for child vertices $x, y \in V(T)$, let $d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}})$ and $d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}})$ with $\tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} = \boldsymbol{b}(q_x)$ be labels with related solutions $\boldsymbol{f}^*_{\boldsymbol{x}}$ and $\boldsymbol{f}^*_{\boldsymbol{y}}$. The combination of the matchings $\boldsymbol{f}^*_{\boldsymbol{x}}$ and $\boldsymbol{f}^*_{\boldsymbol{y}}$ results in a feasible solution $\boldsymbol{f}_{\boldsymbol{v}} := \boldsymbol{f}^*_{\boldsymbol{x}} + \boldsymbol{f}^*_{\boldsymbol{y}}$ to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ problem with degree use $\tilde{\boldsymbol{s}}_{\boldsymbol{v}} := \tilde{\boldsymbol{s}}_{\boldsymbol{x}}$ at origin o_v and $\tilde{\boldsymbol{t}}_{\boldsymbol{v}} := \tilde{\boldsymbol{t}}_{\boldsymbol{y}}$ at target q_v and weight budget $\tilde{\boldsymbol{c}}_{\boldsymbol{v}} := \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}$. We obtain

$$d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \ge d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) = d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}),$$

where $d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}})$ is the label corresponding to vertex $v \in V(T)$. This implies

$$d_v(\tilde{\boldsymbol{s}}_{\boldsymbol{v}}, \tilde{\boldsymbol{t}}_{\boldsymbol{v}}, \tilde{\boldsymbol{c}}_{\boldsymbol{v}}) \le \min_{\substack{\tilde{\boldsymbol{s}}_{\boldsymbol{x}} = \tilde{\boldsymbol{s}}_{\boldsymbol{v}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{y}} = \tilde{\boldsymbol{t}}_{\boldsymbol{v}} \\ \tilde{\boldsymbol{t}}_{\boldsymbol{x}} + \tilde{\boldsymbol{s}}_{\boldsymbol{y}} = \boldsymbol{b}(q_x) \\ \tilde{\boldsymbol{c}}_{\boldsymbol{x}} + \tilde{\boldsymbol{c}}_{\boldsymbol{y}} = \tilde{\boldsymbol{c}}_{\boldsymbol{v}}}} \left\{ d_x(\tilde{\boldsymbol{s}}_{\boldsymbol{x}}, \tilde{\boldsymbol{t}}_{\boldsymbol{x}}, \tilde{\boldsymbol{c}}_{\boldsymbol{x}}) + d_y(\tilde{\boldsymbol{s}}_{\boldsymbol{y}}, \tilde{\boldsymbol{t}}_{\boldsymbol{y}}, \tilde{\boldsymbol{c}}_{\boldsymbol{y}}) \right\}.$$

Finally, we obtain an optimal robust perfect $\boldsymbol{b}$-matching in an SP graph by backtracking the steps of the DP and considering the label associated to the SP tree's root as shown in the following lemma.

Lemma 6.4. *Let $r \in V(T)$ be the root of SP tree T and let $\boldsymbol{f}$ be an optimal robust perfect $\boldsymbol{b}$-matching in SP graph G. The weight of robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f}$ is*

$$c(\boldsymbol{f}) = \min \Big\{ \hat{c} \;\Big|\; \exists\; \tilde{\boldsymbol{c}}_{\boldsymbol{r}} \in \{0,\dots,C_r\}^k : \max_{\lambda \in [k]} \tilde{c}_r^{\lambda} = \hat{c} \;\wedge\; d_r(\boldsymbol{b}(o_r), \boldsymbol{b}(q_r), \tilde{\boldsymbol{c}}_{\boldsymbol{r}}) = 0 \Big\} \tag{9}$$

with $C_r = B_r \sum_{e \in E(G_r)} c(e)$ *where* $B_r = \max_{\lambda \in [k], v \in V(G_r)} b^{\lambda}(v)$.

Proof. First, we note that by definition of the root vertex $r \in V(T)$ it holds $G_r = G$, $o_r = o$, and $q_r = q$ where o and q denote the origin and target of graph G, respectively. For all vertices $v \in V(G)$, the degree constraints of the RobPM≡ problem are ensured by constraints (5) of the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{r}}, \tilde{\boldsymbol{t}}_{\boldsymbol{r}}, \tilde{\boldsymbol{c}}_{\boldsymbol{r}})$ problem with $\tilde{\boldsymbol{s}}_{\boldsymbol{r}} = \boldsymbol{b}(o_r)$ and $\tilde{\boldsymbol{t}}_{\boldsymbol{r}} = \boldsymbol{b}(q_r)$. The consistent selection and capacity constraints as well as the integer conditions of the RobPM≡ problem are one to one included in the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{r}}, \tilde{\boldsymbol{t}}_{\boldsymbol{r}}, \tilde{\boldsymbol{c}}_{\boldsymbol{r}})$ problem by constraints (6), (7), and (8), respectively. Accordingly, every feasible solution to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{r}}, \tilde{\boldsymbol{t}}_{\boldsymbol{r}}, \tilde{\boldsymbol{c}}_{\boldsymbol{r}})$ problem is also a feasible solution to the RobPM≡ problem.

However, the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{r}}, \tilde{\boldsymbol{t}}_{\boldsymbol{r}}, \tilde{\boldsymbol{c}}_{\boldsymbol{r}})$ problem includes one additional set of constraints, namely constraints (4). Constraints (4) control whether the weight of a matching is equal to the weight budget. For this reason, we look for a budget $\tilde{\boldsymbol{c}}_{\boldsymbol{r}} \in \{0,\dots,C_r\}^k$ for which a feasible solution to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{r}}, \tilde{\boldsymbol{t}}_{\boldsymbol{r}}, \tilde{\boldsymbol{c}}_{\boldsymbol{r}})$ problem exists, i.e., for which $d_r(\boldsymbol{b}(o_r), \boldsymbol{b}(q_r), \tilde{\boldsymbol{c}}_{\boldsymbol{r}}) = 0$ holds true. A feasible solution to the rRobPM≡$(\tilde{\boldsymbol{s}}_{\boldsymbol{r}}, \tilde{\boldsymbol{t}}_{\boldsymbol{r}}, \tilde{\boldsymbol{c}}_{\boldsymbol{r}})$ problem corresponds to a robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f}$ with weight $c(\boldsymbol{f}) = \max_{\lambda \in [k]} c(f^{\lambda}) = \max_{\lambda \in [k]} \tilde{c}_r^{\lambda}$. Therefore, the weight of an optimal robust perfect $\boldsymbol{b}$-matching is the minimum maximum budget needed among all scenarios $\lambda \in [k]$ which we obtain by expression (9).

Overall, we obtain the pseudo-polynomial-time solvability of a special case of the RobPM≡ problem on SP graphs as shown in the following theorem.

Theorem 6.3. *Let $(G, u, c, \boldsymbol{b})$ be a* RobPM≡ *instance where G is an SP graph. The DP described computes an optimal robust perfect $\boldsymbol{b}$-matching in pseudo-polynomial time if the number of scenarios is constant.*

Proof. The correctness of the DP results from Lemmas 6.1-6.4. Considering the runtime, we first note that the representation of an SP graph G by its SP tree T can be computed in $\mathcal{O}(|E(G)|)$ time [38]. At every SP tree's vertex $v \in V(T)$ labels for all degree uses $\tilde{\boldsymbol{s}}_{\boldsymbol{v}}$, $\tilde{\boldsymbol{t}}_{\boldsymbol{v}}$ and weight budgets $\tilde{\boldsymbol{c}}_{\boldsymbol{v}}$ need to be calculated where the number of combinations is limited by $(B+1)^{2k} \cdot (BC+1)^k$ with $B = \max_{\lambda \in [k], v \in V(G)} b^{\lambda}(v)$ and $C = \sum_{e \in E(G)} c(e)$. As SP tree T of SP graph G has exactly $|V(T)| = 2|E(G)| - 1$ vertices, we have to compute at most $2|E(G)| \cdot (B+1)^{2k} \cdot (BC+1)^k$ labels. It remains to bound the complexity for computing the labels. If $v \in V(T)$ is an L-vertex, we can compute the corresponding label in $\mathcal{O}(1)$ time. If $v \in V(T)$ is an S- or P-vertex, we need to compute the minimum of less than $(B+1)^{2k}(BC+1)^k$ sums which is in $\mathcal{O}((B+1)^{2k}(BC+1)^k)$. In total, we obtain a runtime of $\mathcal{O}(2|E(G)| \cdot (B+1)^{4k} \cdot (BC+1)^{2k})$.

By Theorem 6.3, we obtain a special case solvable in polynomial time if, in addition to the constant number of scenarios, $B \in \mathcal{O}(|E(G)|)$ and $C \in \mathcal{O}(|E(G)|)$ hold. For instance, we note that a feasible solution to the RobPM≡ problem on SP graphs with a constant number of scenarios and zero-one degree restrictions is determined in polynomial time as we can neglect the weight or just set $c \equiv 1$. Finally, we summarize the results for the RobPM≡ problem on SP graphs in the following corollary.

Corollary 6.2. *The* RobPM≡ *problem is weakly $\mathcal{NP}$-hard on SP graphs, even if only two scenarios are considered with zero-one degree restrictions. If the number of scenarios is not part of the input, the* RobPM≡ *problem is solvable in pseudo-polynomial time. If in addition $B \in \mathcal{O}(|E(G)|)$ and $C \in \mathcal{O}(|E(G)|)$ hold, the* RobPM≡ *problem is solvable in polynomial time.*

Complexity for pearl graphs. In this section, we analyze the complexity of the RobPM≡ problem on SP graphs with a specific structure—the so-called pearl graphs. Using the representation of SP trees, we define pearl graphs based on the definition of Ohst [34].

Definition 6.3. *A pearl graph is an SP graph where the corresponding SP tree does not have P-vertices whose child vertices are S-vertices.*

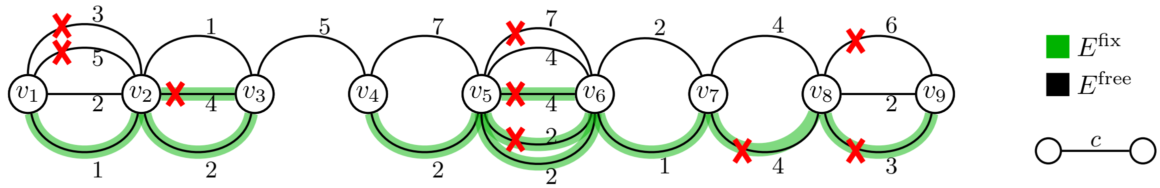

Figure 9: Deletion of needless edges

In other words, a pearl graph is a path consisting of multi-edges termed pearls. We note that the result of Corollary 6.1 can be formulated stronger as the reduction is based on a pearl graph. We obtain that the RobPM≡ problem is $\mathcal{NP}$-hard on pearl graphs, even if only two scenarios are considered with unit edge capacities.

In the following, we analyze the complexity of the RobPM≡ problem on pearl graphs if only zero-one degree restrictions are given. Using the special structure of a pearl graph and the zero-one degree restrictions, we may reduce each pearl to either the lightest fixed and free edge or the lightest free edge only (provided the edges exist) as visualized in Figure 9. If an optimal robust perfect $\boldsymbol{b}$-matching did not match either the lightest fixed or the lightest free edge of each pearl, the weight could be reduced. Furthermore, a fixed edge with higher or equal weight compared to a free edge of the same incident vertices can be neglected because of the restrictive consistent selection constraints for the fixed edge. Using this reduced graph, we present the following lemma.

Lemma 6.5. *Let $G = (V, E = E^{\text{fix}} \cup E^{\text{free}})$ be a peal graph where each pearl consists of at most one fixed and one free edge. Let $\mathcal{I} = (G, u, c, \boldsymbol{b})$ be a corresponding* RobPM≡ *instance with zero-one degree restrictions $b^\lambda : V \to \{0, 1\}$, $\lambda \in [k]$. Furthermore, let a minimum weight perfect b^λ-matching f^λ be given for all scenarios $\lambda \in [k]$. If a robust perfect $\boldsymbol{b}$-matching exists, a minimum weight robust perfect $\boldsymbol{b}$-matching $\tilde{\boldsymbol{f}} = (\tilde{f}^1, \ldots, \tilde{f}^k)$ is defined as follows*

$$\tilde{f}^\lambda(e) = \begin{cases} 1 & \text{for } e \in E^{\text{fix}} \text{ if } f^{\lambda'}(e) = 1 \text{ for all } \lambda' \in [k], \\ 1 & \text{for } e = \{v, w\} \in E^{\text{free}} \text{ if } e^{\text{fix}} = \{v, w\} \in E^{\text{fix}} \text{ with } f^\lambda(e^{\text{fix}}) = 1 \text{ and} \\ & f^{\lambda'}(e^{\text{fix}}) = 0 \text{ for one } \lambda' \in [k], \\ 1 & \text{for } e \in E^{\text{free}} \text{ if } f^\lambda(e) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Firstly, we note that a perfect b^λ-matching is uniquely defined on a path (if a matching exists) for all scenarios $\lambda \in [k]$. Consequently, the vertices that are matched with each other are uniquely defined on a pearl graph for each scenario $\lambda \in [k]$. However, for an optimal robust perfect $\boldsymbol{b}$-matching, it remains to determine which edge of a multi-edge matches each two vertices for all scenarios.

Let f^λ be a minimum weight perfect b^λ-matchings for scenario $\lambda \in [k]$. If matchings $f^1, \ldots, f^k$ satisfy the consistent selection constraints, we obtain a minimum weight robust perfect $\boldsymbol{b}$-matching by $\boldsymbol{f} = (f^1, \ldots, f^k)$. Otherwise, we determine a minimum weight robust perfect $\boldsymbol{b}$-matching as follows. For all scenarios $\lambda \in [k]$, we use b^λ-matching f^λ to determine which vertices need to be matched with each other. For each two vertices $u, v \in V$ that need to be matched with each other in one scenario $\lambda \in [k]$, it holds $f^\lambda(e) = 1$ for $e = \{u, v\} \in E$. If $e \in E^{\text{fix}}$, the consistent selection constraints need to be satisfied, i.e., $f^{\lambda'}(e) = 1$ for all $\lambda' \in [k]$. If the consistent selection constraints are not satisfied, i.e., $f^{\lambda'}(e) = 0$ for at least one $\lambda' \in [k]$, vertices u, v need to be matched by a different, free edge $e' = \{u, v\} \in E^{\text{free}}$. If $e \in E^{\text{free}}$, there exists no lighter fixed edge incident to vertices u, v as f^λ is a minimum weight perfect b^λ-matching. As a result, we obtain an optimal robust perfect $\boldsymbol{b}$-matching as stated in the lemma if one exists.

According to Lemma 6.5, we can compute a minimum weight robust perfect $\boldsymbol{b}$-matching on pearl graphs in polynomial time if only zero-one degree restrictions are given. We note that the polynomial-time solvability is given independent of the number of scenarios and the weight.

Complexity for cactus graphs. In this paragraph, we analyze the complexity of the RobPM≡ problem on cactus graphs defined as follows.

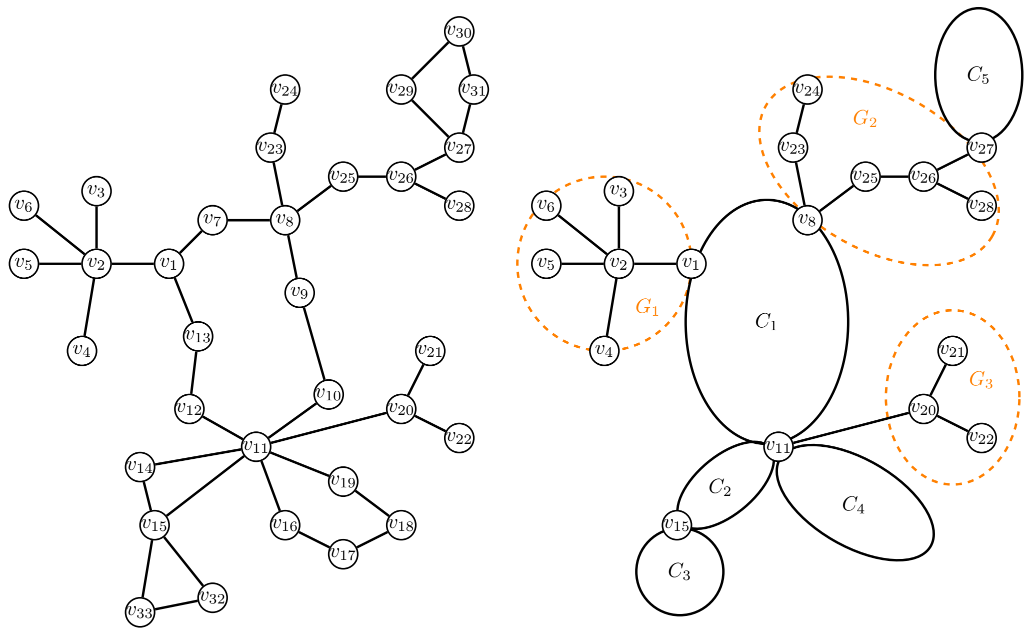

Figure 10: Example of a cactus graph and its skeleton consisting of grafts G_1, G_2, G_3 and cycles C_1, C_2, C_3, C_4, C_5 connected by hinges v_1, v_8, v_{11}, v_{15}, v_{27}

Definition 6.4. *A cactus graph is a graph in which all cycles are pairwise edge disjoint.*

In other words, a cactus graph is a graph in which two cycles have at most one vertex in common. In the following, we consider the so-called *skeleton* of a cactus graph based on the definition of Burkard and Krarup [10]. A skeleton of a cactus graph indicates the composition of the graph by two different blocks, namely grafts and cycles, connected by so-called hinges as visualized in Figure 10. A *hinge* is a vertex of degree at least three that is included in at least one cycle, also termed as H-vertex. A *cycle* consists of so-called C-vertices, which are vertices of degree two, and H-vertices. A *graft* is a maximal subtree induced only by G-vertices, which are vertices that are not included in a cycle, and H-vertices. The skeleton of a cactus graph is represented by a rooted tree consisting of a vertex for each of the grafts, cycles, and hinges as visualized in Figure 11. We note that a block is to be chosen as the root.

In general, the RobPM≡ problem is weakly $\mathcal{NP}$-hard on cactus graphs as shown in the following theorem.

Theorem 6.4. *The* RobPM≡ *problem is weakly $\mathcal{NP}$-hard on cactus graphs, even if only two scenarios are considered.*

Proof. The RobPM≡ problem is included in $\mathcal{NP}$ as we can check in polynomial time if the capacity constraints, the degree constraints with equality, and the consistent selection constraints are satisfied for all scenarios. We perform a reduction from the weakly $\mathcal{NP}$-complete partition problem [22]. Let $\mathcal{I}$ be a partition instance defined by n positive integers $a_1, \ldots, a_n$ that sum up to $\sum_{i=1}^n a_i = 2w$. Partition asks whether there exists a partition of set A into two disjoint subsets A_1 and A_2 such that the sum of the integers of each subset is equal, i.e., $w = \sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$.

We construct a RobPM≡ instance $\widetilde{\mathcal{I}}$ as visualized in Figure 12. The instance is based on a graph $G = (V, E)$ defined as follows. Vertex set V contains two auxiliary vertices v_1^0, v_2^0 and seven vertices $v_1^i, \ldots, v_7^i$ per integer a_i, $i \in [n]$. Edge set E includes edges that connect the auxiliary vertices by $e^0 = \{v_1^0, v_2^0\}$ and $e_0^1 = \{v_2^0, v_1^1\}$. Furthermore, edge set E includes edges that connect each seven vertices by $e_j^i = \{v_j^i, v_{(j+1)}^i\}$ for $j \in [6]$, $i \in [n]$ as well as $\tilde{e}^i = \{v_2^i, v_4^i\}$, $\hat{e}^i = \{v_5^i, v_7^i\}$, and $\bar{e}^i = \{v_4^i, v_5^i\}$ for all $i \in [n]$. Finally, edge set E includes the edges $e_0^{i+1} = \{v_7^i, v_1^{i+1}\}$ for $i \in [n-1]$. The fixed edges of set E are defined by the edges $\bar{e}^i$ for all $i \in [n]$ contained in set E^{fix}. The remaining edges are contained in set E^{free}. We define the capacity u and the weight c on all edges $e \in E$

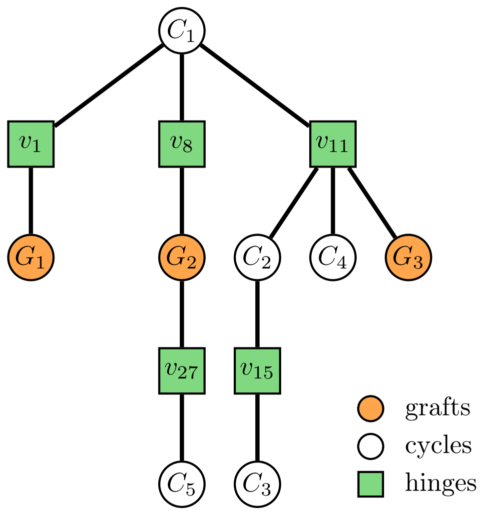

Figure 11: Example of the representation of a skeleton of a cactus graph by a rooted tree

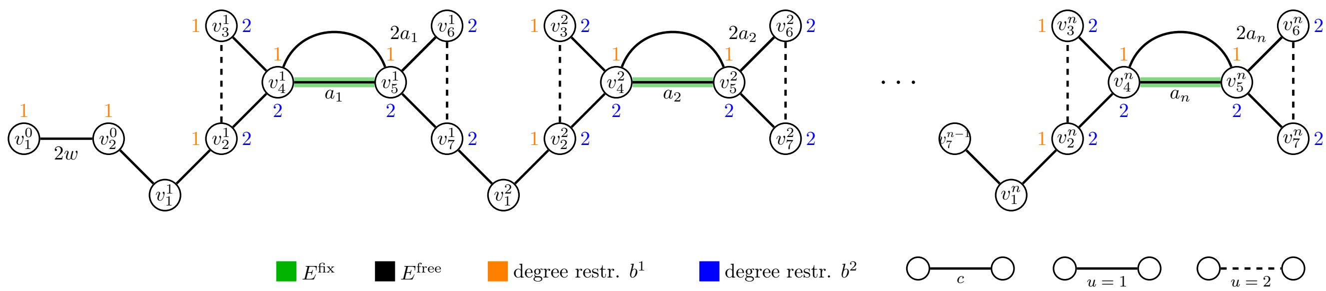

Figure 12: Construction of RobPM≡ instance $\widetilde{\mathcal{I}}$

by

$$u(e) = \begin{cases} 2 & \text{if } e = e_2^i \text{ or } e = e_6^i, \\ 1 & \text{otherwise,} \end{cases} \qquad c(e) = \begin{cases} 2w & \text{if } e = e^0, \\ 2a_i & \text{if } e = e_5^i,\ i \in [n], \\ a_i & \text{if } e = \bar{e}^i,\ i \in [n], \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, we define the degree restrictions $\boldsymbol{b} = (b^1, b^2)$ on all vertices $v \in V$ by

$$b^1(v) = \begin{cases} 1 & \text{for all } v \in \{v_1^0, v_2^0\} \cup \{v_j^i \mid j \in \{2, \dots, 5\}, i \in [n]\}, \\ 0 & \text{otherwise,} \end{cases}$$

$$b^2(v) = \begin{cases} 2 & \text{if } v \in \{v_j^i \mid j \in \{2, \dots, 7\}, i \in [n]\}, \\ 0 & \text{otherwise.} \end{cases}$$

Overall, we obtain a feasible RobPM≡ instance $\widetilde{\mathcal{I}}$ constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance if and only if for instance $\widetilde{\mathcal{I}}$ a robust perfect $\boldsymbol{b}$-matching exists with weight of at most $\beta := 3w$.

Let A_1, A_2 be a feasible partition for instance $\mathcal{I}$. For the first scenario, we define matching f^1 by

$$f^1(e) = \begin{cases} 1 & \text{for } e = e^0 \text{ and } e_2^i, \\ 1 & \text{for } e = \bar{e}^i \text{ if } a_i \in A_1, \\ 1 & \text{for } e = e_4^i \text{ if } a_i \in A_2, \\ 0 & \text{otherwise.} \end{cases}$$

The weight is

$$\begin{aligned}c(f^1) &= \sum_{e\in E} c(e)f^1(e) = c(e^0)f^1(e^0) + \sum_{i\in[n]:\ a_i\in A_1} c(\bar{e}^i)f^1(\bar{e}^i) + \sum_{i\in[n]:\ a_i\in A_2} c(e_4^i)f^1(e_4^i)\\ &= 2w + \sum_{a_i\in A_1} a_i + \sum_{a_i\in A_2} 0 = 3w.\end{aligned}$$

For the second scenario, we define matching f^2 by

$$f^2(e) = \begin{cases} 2 & \text{for } e\in\{e_2^i, e_6^i\} \text{ if } a_i\in A_1,\\ 1 & \text{for } e\in\{e_4^i, \bar{e}^i\} \text{ if } a_i\in A_1,\\ 1 & \text{for } e\in\{e_2^i, e_3^i, e_5^i, e_6^i\} \text{ and } e\in\{\tilde{e}^i, \hat{e}^i\} \text{ if } a_i\in A_2,\\ 0 & \text{otherwise.}\end{cases}$$

The weight is

$$\begin{aligned}c(f^2) &= \sum_{e\in E} c(e)f^2(e)\\ &= \sum_{i\in[n]:\ a_i\in A_1}\left(\sum_{j\in\{2,4,6\}} c(e_j^i)f^2(e_j^i) + c(\bar{e}^i)f^2(\bar{e}^i)\right) + \sum_{i\in[n]:\ a_i\in A_2}\left(\sum_{j\in\{2,3,5,6\}} c(e_j^i)f^2(e_j^i) + \sum_{e\in\{\tilde{e}^i,\hat{e}^i\}} c(e)f^2(e)\right)\\ &= \sum_{a_i\in A_1}(0+a_i) + \sum_{a_i\in A_2}(2a_i+0) = 3w.\end{aligned}$$

Consequently, we have constructed a robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f} = (f^1, f^2)$ with weight of $3w$.

Conversely, let $\boldsymbol{f} = (f^1, f^2)$ be a robust perfect $\boldsymbol{b}$-matching with weight $c(\boldsymbol{f}) = \max\{c(f^1), c(f^2)\} \le 3w$. Let I_{fix} denote the set of indices that indicate which of the fixed edges $\bar{e}^1, \ldots, \bar{e}^n$ are matched in robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f} = (f^1, f^2)$, i.e., $I_{\text{fix}} = \{i\in[n] \mid f^1(\bar{e}^i) = f^2(\bar{e}^i) = 1\}$. Perfect b^1-matching f^1 covers vertices v_1^0, v_2^0 and v_j^i for all $j\in\{2,\ldots,5\}$ and $i\in[n]$. We note that the coverage of vertices v_1^0 and v_2^0 contributes $2w$ to the weight. A lower bound on the weight is given by

$$c(f^1) = \sum_{e\in E} c(e)f^1(e) \ge c(e^0) + \sum_{i\in I_{\text{fix}}} c(\bar{e}^i) = 2w + \sum_{i\in I_{\text{fix}}} a_i$$

As $c(f^1)\le 3w$ holds, we obtain $\sum_{i\in I_{\text{fix}}} a_i \le w$. Perfect b^2-matching f^2 covers all vertices with a value of two except the auxiliary vertices v_1^0, v_2^0 and vertices v_1^i for all $i\in[n]$. For each seven integer vertices, there exist only two different perfect matchings due to the capacity constraints. Either edges e_2^i, e_6^i and edges e_4^i, $\bar{e}^i$ are matched with values two and one, respectively, or edges e_2^i, e_3^i, e_5^i, e_6^i, $\tilde{e}^i$, and $\hat{e}^i$ are matched with value one. We note that only the first option includes a fixed edge. A lower bound on the weight is given by

$$\begin{aligned}c(f^2) &= \sum_{e\in E} c(e)f^2(e) \ge \sum_{i\in I_{\text{fix}}} c(\bar{e}^i) + \sum_{i\in[n]\setminus I_{\text{fix}}} c(e_5^i) = \sum_{i\in I_{\text{fix}}} a_i + \sum_{i\in[n]\setminus I_{\text{fix}}} 2a_i = 2\sum_{i=1}^n a_i - \sum_{i\in I_{\text{fix}}} a_i\\ &= 4w - \sum_{i\in I_{\text{fix}}} a_i\end{aligned}$$

As $c(f^2)\le 3w$ holds, we obtain $\sum_{i\in I_{\text{fix}}} a_i \ge w$. Overall, we obtain $\sum_{i\in I_{\text{fix}}} a_i = w$. We define the sets $A_1 = \{a_i \mid i\in I_{\text{fix}}\}$ and $A_2 = \{a_i \mid i\in[n]\setminus I_{\text{fix}}\}$ which form a feasible partition.

In the next step, we analyze the complexity of the RobPM≡ problem on cactus graphs for the special case of zero-one degree restrictions. For this purpose, we initially consider the problem on the cactus graphs' blocks. First, we consider the RobPM≡ problem on cycles if only zero-one degree restrictions are given. To determine a robust perfect $\boldsymbol{b}$-matching on a cycle, there must exist feasible perfect b^λ-matchings for all scenarios $\lambda\in[k]$ that satisfy the consistent selection constraints. If there exists one scenario $\lambda'\in[k]$ in which one vertex has degree restriction of zero, $b^{\lambda'}$-matching $f^{\lambda'}$ is uniquely defined. This holds true as such a cycle can be reduced to a path without this vertex, on which a $b^{\lambda'}$-matching is uniquely defined. Consequently, the b^{λ}-matchings of the remaining scenarios $\lambda \in [k]$, $\lambda \neq \lambda'$ are also uniquely defined due to the fixed edges. As a result, a robust perfect $\boldsymbol{b}$-matching is uniquely defined if one exists. We note that there only exist two different feasible robust perfect $\boldsymbol{b}$-matchings on a cycle if the cycle consists of an even number of vertices that have degree restrictions of one in all scenarios. We obtain the following corollary.

Corollary 6.3. *The* RobPM≡ *problem is solvable in polynomial time on cycles if only zero-one degree restrictions are given.*

Clearly, we also obtain the polynomial-time solvability of the RobPM≡ problem with zero-one degree restrictions if the graph consists of several disconnected cycles. Interestingly, the RobPM≡ problem becomes weakly $\mathcal{NP}$-hard as soon as two adjacent vertices of the cycle are connected by an additional parallel edge as shown in the following lemma. A proof can be found in Appendix B. However, we note that the definition of cactus graphs does not allow parallel edges on cycles.

In the next step, we analyze the complexity of the RobPM≡ problem on trees if only zero-one degree restrictions are given. To determine a robust perfect $\boldsymbol{b}$-matching on trees, there must exist feasible perfect b^{λ}-matchings for all scenarios $\lambda \in [k]$ that satisfy the consistent selection constraints. Considering a single scenario $\lambda \in [k]$, we note that a perfect b^{λ}-matching is uniquely defined on a tree (if one exists). We obtain the only perfect b^{λ}-matching iteratively starting from a leaf of the tree. Let $e = \{u, v\}$ denote the incident edge of a leaf u. If $b^{\lambda}(u) = 1$ and $b^{\lambda}(v) = 1$, we add edge e to the matching and delete it from the tree. If $b^{\lambda}(u) = 0$ and $b^{\lambda}(v) = 1$, we delete edge e from the tree. If $b^{\lambda}(u) = 1$ and $b^{\lambda}(v) = 0$, we stop as there does not exist a feasible perfect b^{λ}-matching. For each scenario $\lambda \in [k]$, this procedure results in the only optimal perfect b^{λ}-matching as there do not exist cycles in trees. As a result, a robust perfect $\boldsymbol{b}$-matching is uniquely defined if one exists. We obtain the following corollary.

Corollary 6.4. *The* RobPM≡ *problem is solvable in polynomial time on trees if only zero-one degree restrictions are given.*

Finally, we analyze the complexity of the RobPM≡ problem on cactus graphs if only zero-one degree restrictions are given. Considering a single scenario $\lambda \in [k]$, we note that a perfect b^{λ}-matching is uniquely defined on a cactus graph except for the cycles with an even number of vertices that have degree restrictions of one (if a matching exists). We obtain a partly uniquely defined perfect b^{λ}-matching by constructing the matching starting from a leaf of the skeleton. Even if the leaf corresponds to a cycle, it is uniquely defined whether the only included hinge of the leaf is matched with another vertex of the leaf or within an adjacent block. Subsequently, we can determine whether the remaining hinges of the adjacent block are matched within or beyond the block. Similar to the above procedure on trees for determining matchings, we proceed to construct a matching along the skeleton from bottom up. For each scenario $\lambda \in [k]$, this procedure results in a perfect b^{λ}-matching that is uniquely defined except on cycles with an even number of vertices that have degree restrictions of one. More importantly, we obtain in which blocks the hinges are matched. Using this, we present Algorithm 6.1 to solve the RobPM≡ problem on cactus graphs if only zero-one degree restrictions are given.

Algorithm 6.1

Input: $\textsc{RobPM}\equiv$ instance $(G, u, c, \boldsymbol{b})$ where $G = (V, E = E^{\text{fix}} \cup E^{\text{free}})$ is a cactus graph and $b^\lambda : V \to \{0, 1\}$, $\lambda \in [k]$

Output: Minimum weight robust perfect $\boldsymbol{b}$-matching

Initialization: $\widetilde{E}^{\text{fix}} = E^{\text{fix}}$

Method:

for all scenarios $\lambda \in [k]$ **do**
    Determine a minimum weight perfect b^λ-matching f^λ
while $\widetilde{E}^{\text{fix}} \neq \emptyset$ **do**
    Choose an edge $\tilde{e} \in \widetilde{E}^{\text{fix}}$
    if the consistent selection constraints are not satisfied on an edge $\tilde{e}$ **then**
        if $\tilde{e}$ is included in a cycle C **then**
            Consider the $\textsc{RobPM}\equiv$ instance $\mathcal{I}' = (C, u, c, \bar{\boldsymbol{b}})$ restricted to cycle C with new degree restrictions $\bar{\boldsymbol{b}} = (\bar{b}^1, \dots, \bar{b}^k)$ defined by

$$\bar{b}^\lambda(v) = \begin{cases} b^\lambda(v) & \text{if } v \in V(C) \text{ is a } C\text{-vertex} \\ \sum\limits_{e=\{v,u\} \in E(C)} f^\lambda(e) & \text{if } v \in V(C) \text{ is a } H\text{-vertex} \end{cases}$$

            Compute an optimal robust perfect $\bar{\boldsymbol{b}}$-matching $\bar{\boldsymbol{f}}$ for instance $\mathcal{I}'$
            Set $f^\lambda(e) = \bar{f}^\lambda(e)$ for all $\lambda \in [k]$ and $e \in E(C)$
            Set $\widetilde{E}^{\text{fix}} = \widetilde{E}^{\text{fix}} \setminus E(C)$
        else stop
    else
        Set $\widetilde{E}^{\text{fix}} = \widetilde{E}^{\text{fix}} \setminus \{\tilde{e}\}$
return robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f} = (f^1, \dots, f^k)$

We conclude this section with the polynomial-time solvability of the $\textsc{RobPM}\equiv$ problem on cactus graphs if only zero-one degree restrictions are given as shown in the following theorem.

Theorem 6.5. *Let $\mathcal{I} = (G, u, c, \boldsymbol{b})$ be a $\textsc{RobPM}\equiv$ instance where G is a cactus graph with zero-one degree restrictions $b^\lambda : V \to \{0, 1\}$, $\lambda \in [k]$. Algorithm 6.1 computes an optimal robust perfect $\boldsymbol{b}$-matching in polynomial time.*

Proof. Algorithm 6.1 first determines b^λ-matchings for all scenarios $\lambda \in [k]$. Subsequently, the consistent selection constraints are checked on all fixed edges. We note that every (fixed) edge is included either in a graft or a cycle. For this reason, we distinguish between two cases.

First, if the consistent selection constraints are not satisfied on an edge included in a cycle C, a robust perfect $\bar{\boldsymbol{b}}$-matching is computed on the instance $\mathcal{I}' = (C, u, c, \bar{\boldsymbol{b}})$ restricted to cycle C with new degree restrictions $\bar{\boldsymbol{b}}$. The new degree restrictions $\bar{\boldsymbol{b}}$ result by updating the degree restrictions $\boldsymbol{b}$ at the hinges of cycle C on the basis of matching $\boldsymbol{f}$. More precisely, if a perfect b^λ-matching f^λ, $\lambda \in [k]$ covers a hinge beyond cycle C, the degree restriction $\bar{b}^\lambda$ of the hinge is set to zero. If there exists an optimal robust perfect $\bar{\boldsymbol{b}}$-matching $\bar{\boldsymbol{f}}$ for instance $\mathcal{I}'$, the robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f}$ is replaced on cycle C, i.e., $f^\lambda(e) = \bar{f}^\lambda(e)$ for all $\lambda \in [k]$ and $e \in E(C)$. Furthermore, all fixed edges of the cycle may be deleted from the set of fixed edges that need to be checked for the consistent selection constraints.

Second, if the consistent selection constraints are not satisfied on an edge included in a graft, there does not exist a robust perfect $\boldsymbol{b}$-matching on the cactus graph.

Assume the robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f}$ computed by the algorithm is not optimal. There exists a robust perfect $\boldsymbol{b}$-matching $\tilde{\boldsymbol{f}}$ with less weight, i.e., $c(\tilde{\boldsymbol{f}}) < c(\boldsymbol{f})$. We note that the robust perfect $\boldsymbol{b}$-matching $\tilde{\boldsymbol{f}}$ differs from the robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f}$ in at least one scenario. In the following, we show that the symmetric difference $\tilde{\boldsymbol{f}} \Delta \boldsymbol{f}$ forms cycles consisting of an even number of vertices of degree restriction one. We note that matchings $\tilde{\boldsymbol{f}}$ and $\boldsymbol{f}$ are equal on all grafts of the cactus graph as the blocks in which the hinges are matched as well as matchings on trees are uniquely defined in all scenarios. Accordingly, the symmetric difference $\tilde{\boldsymbol{f}}\Delta\boldsymbol{f}$ may only include edges of cycles. However, the symmetric difference $\tilde{\boldsymbol{f}}\Delta\boldsymbol{f}$ may not include edges of a cycle with an odd number of vertices or of a cycle which includes one vertex of degree restriction zero. If this was not true, the matchings on the corresponding cycles would not be uniquely defined in these two cases. Consequently, the symmetric difference $\tilde{\boldsymbol{f}}\Delta\boldsymbol{f}$ only contains edges of cycles with an even number of vertices of degree restriction one for all scenarios. If the symmetric difference $\tilde{\boldsymbol{f}}\Delta\boldsymbol{f}$ includes one edge of such a cycle, it must also contain all remaining edges of this cycle. This holds true as there only exist two feasible matchings on such a cycle where the matchings are disjoint. Considering the cycles of the symmetric difference $\tilde{\boldsymbol{f}}\Delta\boldsymbol{f}$, we are given that the weight of matching $\boldsymbol{f}$ is higher than or equal to the weight of matching $\tilde{\boldsymbol{f}}$ because of the assumption. We obtain a contradiction as the algorithm ensures an optimal robust perfect matching on a cycle either by minimum weight perfect b^λ-matchings $f^1, \ldots, f^k$ or if necessary by recomputation of a robust matching.

Considering the runtime of the algorithm, we obtain the following. We determine minimum weight perfect b^λ-matchings f^λ for all scenarios in $\mathcal{O}(k|E(G)|)$ time. For all fixed edges, we check if the consistent selection constraints are satisfied in $\mathcal{O}(k|E(G)|)$ time. If necessary, we additionally check whether the fixed edges are included in a cycle, for example, by means of breadth-first search in $\mathcal{O}(|V(G)| \cdot |E(G)| + |E(G)|^2)$ time. For all cycles on which the consistent selection constraints are violated, we compute a minimum weight robust perfect matching in $\mathcal{O}(k|E(G)|)$ time. Overall, a minimum weight perfect $\boldsymbol{b}$-matching is determined in $\mathcal{O}((|V(G)| + k) \cdot |E(G)| + |E(G)|^2)$ time.

7. Conclusion

In this paper, we considered a robust two-stage concept for combinatorial optimization problems under discrete demand uncertainties, the so-called Rob2S≡ problem. First, we introduced a general problem definition. Second, we showed that the Rob2S≡ version of several specific combinatorial optimization problems is $\mathcal{NP}$-hard but provided a polynomial-time solvable special case. Third, we considered the Rob2S≡ version of three combinatorial optimization problems: the representative multi-selection, the shortest path, and the minimum weight perfect b-matching problem; we obtained the RobRS≡, the RobSP≡, and the RobPM≡ problem, respectively. For the RobRS≡ problem, we proved the weak $\mathcal{NP}$-hardness in general and the polynomial-time solvability for the special case that only zero or one element is demanded from every subset. For the RobSP≡ problem, we proved that finding a feasible solution is strongly $\mathcal{NP}$-complete, even if only one fixed arc and two scenarios are considered that have the same end vertex. However, for the special case of acyclic digraphs, we presented a polynomial-time algorithm. For the RobPM≡ problem, we proved that finding a feasible solution is strongly $\mathcal{NP}$-complete, even if only two scenarios are considered on bipartite graphs with zero-one degree restrictions. For the special case of SP graphs, we showed the weak $\mathcal{NP}$-hardness and provided a pseudo-polynomial-time algorithm in case of a constant number of scenarios. For the special case of pearl and cactus graphs, we showed the weak $\mathcal{NP}$-hardness and presented polynomial-time algorithm for the case of only zero-one degree restrictions. For the future work, we will consider the polyeder and look for exact algorithms of the RobRS≡, the RobSP≡, and the RobPM≡ problem.

ACKNOWLEDGMENTS

This study was supported by Freigeist-Fellowship of Volkswagen Stiftung.

References

[1] Eduardo Álvarez-Miranda, Valentina Cacchiani, Tim Dorneth, Michael Jünger, Frauke Liers, Andrea Lodi, Tiziano Parriani, and Daniel R. Schmidt. Models and Algorithms for Robust Network Design with Several Traffic Scenarios. In A. Ridha Mahjoub, V. Markakis, I. Milis, and V. Th. Paschos, editors, *ISCO 2012, Revised Selected Papers*, volume 7422, pages 261–272, 2012.

[2] Alper Atamtürk and Muhong Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[3] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical programming*, 99(2):351–376, 2004.

[4] Piotr Berman, Marek Karpinski, and A Scott. Approximation hardness of short symmetric instances of max-3sat. Technical report, 2004.

[5] Dimitris Bertsimas and Constantine Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

[6] Dimitris Bertsimas, Dan A Iancu, and Pablo A Parrilo. Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394, 2010.

[7] Dimitris Bertsimas, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, and Tongxin Zheng. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE transactions on power systems*, 28(1):52–63, 2012.

[8] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1-3):49–71, 2003.

[9] Christoph Buchheim and Jannis Kurtz. Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211–238, 2018.

[10] Rainer E Burkard and Jakob Krarup. A linear algorithm for the pos/neg-weighted 1-median problem on a cactus. *Computing*, 60(3):193–215, 1998.

[11] Christina Büsing. *Recoverable robustness in combinatorial optimization*. Cuvillier Verlag, 2011.

[12] Christina Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

[13] Christina Büsing, Arie MCA Koster, and Sabrina Schmitz. Robust minimum cost flow problem under consistent flow constraints. *Annals of Operations Research*, 312(2):691–722, 2022.

[14] Christina Büsing, Arie MCA Koster, and Sabrina Schmitz. Robust transshipment problem under consistent flow constraints. *arXiv preprint arXiv:2208.13487*, 2022.

[15] Valentina Cacchiani, Michael Jünger, Frauke Liers, Andrea Lodi, and Daniel R Schmidt. Single-commodity robust network design with finite and hose demand sets. *Mathematical Programming*, 157(1):297–342, 2016.

[16] Constantine Constantine Michael Caramanis. *Adaptable optimization: Theory and algorithms*. PhD thesis, Massachusetts Institute of Technology, 2006.

[17] Vladimir G Deineko and Gerhard J Woeginger. Complexity and in-approximability of a selection problem in robust optimization. *4OR*, 11(3):249–252, 2013.

[18] Alexandre Dolgui and Sergey Kovalev. Min–max and min–max (relative) regret approaches to representatives selection problem. *4OR*, 10(2):181–192, 2012.

[19] V. Gabrel, M. Lacroix, C. Murat, and N. Remli. Robust location transportation problems under uncertain demands. *Discrete Applied Mathematics*, 164:100–111, 2014. Combinatorial Optimization.

[20] Marc Goerigk, Adam Kasperski, and Paweł Zieliński. Robust two-stage combinatorial optimization problems under convex second-stage cost uncertainty. *Journal of Combinatorial Optimization*, 43(3):497–527, 2022.

[21] Ruiwei Jiang, Muhong Zhang, Guang Li, and Yongpei Guan. Benders' decomposition for the two-stage security constrained robust unit commitment problem. In *IIE Annual Conference. Proceedings*, page 1. Institute of Industrial and Systems Engineers (IISE), 2012.

[22] David S Johnson and Michael R Garey. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979.

[23] OE Karaşan, MÇ Pinar, and H Yaman. The robust shortest path problem with interval data. *Computers & Operations Research (to appear)*, 2001.

[24] Adam Kasperski, Adam Kurpisz, and Paweł Zieliński. Recoverable robust combinatorial optimization problems. In *Operations Research Proceedings 2012*, pages 147–153. Springer, 2014.

[25] Adam Kasperski, Adam Kurpisz, and Paweł Zieliński. Approximability of the robust representatives selection problem. *Operations Research Letters*, 43(1):16–19, 2015.

[26] Adam Kasperski and Paweł Zieliński. Robust discrete optimization under discrete and interval uncertainty: A survey. In *Robustness analysis in decision aiding, optimization, and analytics*, pages 113–143. Springer, 2016.

[27] Irit Katriel, Claire Kenyon-Mathieu, and Eli Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. *Theoretical Computer Science*, 408(2-3):213–223, 2008.

[28] Bernhard Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*, volume 2. Springer, 2012.

[29] A. M. C. A. Koster, M. Kutschka, and C. Raack. Robust network design: Formulations, valid inequalities, and computations. *Networks*, 61(2):128–149, 2013.

[30] Panos Kouvelis and Gang Yu. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media, 2013.

[31] Sven O Krumke and Clemens Thielen. Minimum cost flows with minimum quantities. *Information processing letters*, 111(11):533–537, 2011.

[32] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[33] Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and online large-scale optimization*, pages 1–27. Springer, 2009.

[34] Jan Peter Ohst. On the construction of optimal paths from flows and the analysis of evacuation scenarios. 2016.

[35] Fernando Ordónez and Jiamin Zhao. Robust capacity expansion of network flows. *Networks: An International Journal*, 50(2):136–145, 2007.

[36] Hans Georg Seedig. Network flow optimization with minimum quantities. In *Operations research proceedings 2010*, pages 295–300. Springer, 2011.

[37] Aurélie Thiele, Tara Terry, and Marina Epelman. Robust linear optimization with recourse. *Rapport technique*, pages 4–37, 2009.

[38] Jacobo Valdes, Robert E Tarjan, and Eugene L Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11(2):298–313, 1982.

[39] Gang Yu and Jian Yang. On the robust shortest path problem. *Computers & Operations Research*, 25(6):457–468, 1998.

[40] Bo Zeng and Long Zhao. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461, 2013.

[41] İhsan Yanıkoğlu, Bram L. Gorissen, and Dick den Hertog. A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3):799–813, 2019.

A. Appendix

In this section, we analyze the complexity of the sPsA problem as shown in the following theorem.

Theorem A.1. *Deciding whether a feasible solution to the* sPsA *problem exists is strongly $\mathcal{NP}$-complete, even if the specified arc set contains only one arc.*

Proof. The sPsA is contained in $\mathcal{NP}$ as we can check in polynomial time whether an (s,t)-path is simple and includes a specified arc. We perform a reduction from the disjoint connecting path problem which is $\mathcal{NP}$-hard, even if the digraph is acyclic and only two commodities are considered. Let $\mathcal{I}$ be a corresponding instance defined by a digraph G and two commodities (s_i, t_i), $i \in [2]$. The disjoint connecting path problem asks if there exist arc disjoint (s_i, t_i)-paths for all $i \in [2]$.

We construct an sPsA instance $\widetilde{\mathcal{I}} = (\widetilde{G}, \tilde{s}, \tilde{t}, \tilde{a})$ as visualized in Figure A.13. We obtain digraph $\widetilde{G}$ by adding

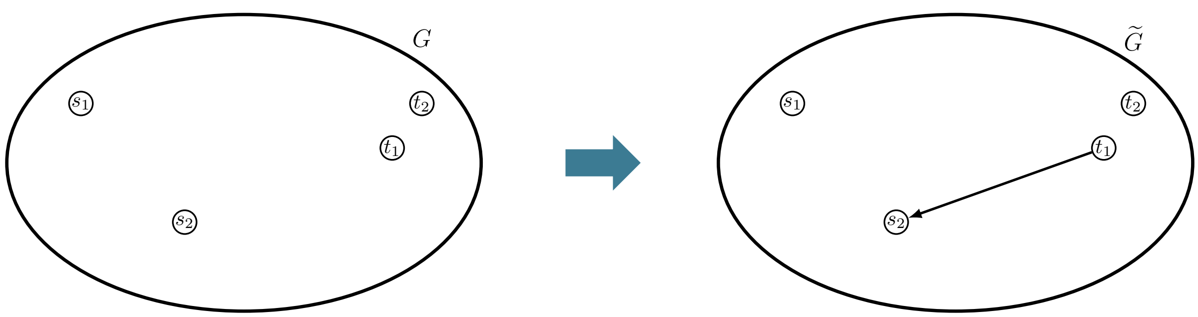

Figure A.13: Transformation from ADP instance $\mathcal{I}$ to sPsA instance $\widetilde{\mathcal{I}}$.

arc $\tilde{a} := (t_1, s_2)$ to digraph G (if it is not already included in G), i.e., $\widetilde{G} = (V, A \cup \{(t_1, s_2)\})$. Furthermore, we set $\tilde{s} = s_1$, $\tilde{t} = t_2$. It remains to show that $\mathcal{I}$ is a Yes-instance if and only if for instance $\widetilde{\mathcal{I}}$ a simple (s,t)-path exists that includes arc a.

Let p_1 be an (s_1, t_1)-path and p_2 be an (s_2, t_2)-path in digraph G that are arc disjoint. Without loss of generality, we assume that paths p_1 and p_2 do not include a cycle, otherwise we delete the cycles. We note that it holds $(t_1, s_2) \notin A(p_1) \cup A(p_2)$. We define an $(\tilde{s}, \tilde{t})$-path $\tilde{p}$ by connecting paths p_1 and p_2 via arc $\tilde{a} = (t_1, s_2)$. Consequently, we have constructed a simple $(\tilde{s}, \tilde{t})$-path $\tilde{p}$ that uses arc $\tilde{a}$.

Conversely, let $\tilde{p}$ be a simple $(\tilde{s}, \tilde{t})$-path in digraph $\widetilde{G}$ that includes arc $\tilde{a}$. Let p_1 denote the subpath of path $\tilde{p}$ from vertex $\tilde{s}$ to the tail of arc $\tilde{a}$. Furthermore, let p_2 denote the subpath of path $\tilde{p}$ from the head of arc $\tilde{a}$ to vertex $\tilde{t}$. The (s_1, t_1)-path p_1 and the (s_2, t_2)-path p_2 are disjoint as $\tilde{p}$ is a simple path.

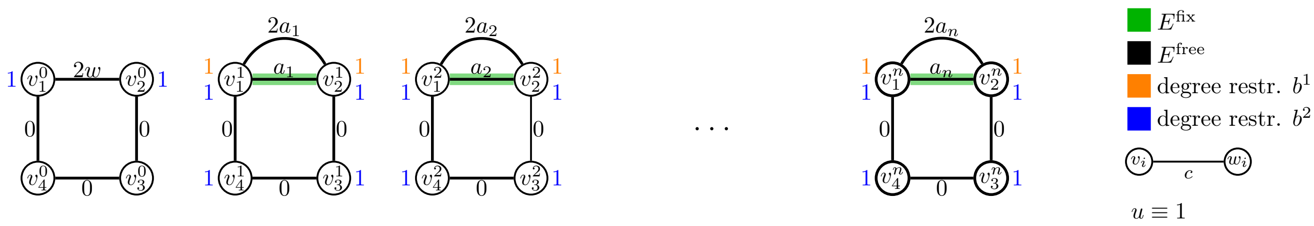

Figure B.14: Construction of RobPM≡ instance $\widetilde{\mathcal{I}}$

B. Appendix

Lemma B.1. *The* RobPM≡ *problem is weakly $\mathcal{NP}$-hard on graphs that consist of cycles with multi-edges, even if only zero-one degree restrictions are considered and each cycle has only one multi-edge.*

Proof. The RobPM≡ problem is included in $\mathcal{NP}$ as we can check in polynomial time if the capacity constraints, the degree constraints with equality, and the consistent selection constraints are satisfied for all scenarios. We perform a reduction from the weakly $\mathcal{NP}$-complete partition problem [22]. Let $\mathcal{I}$ be a partition instance defined by n positive integers $a_1, \dots, a_n$ that sum up to $\sum_{i=1}^{n} a_i = 2w$. Partition asks whether there exists a partition of set A into two disjoint subsets A_1 and A_2 such that the sum of the integers of each subset is equal, i.e., $w = \sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$.

We construct a RobPM≡ instance $\widetilde{\mathcal{I}}$ as visualized in Figure B.14. The instance is based on a graph $G = (V, E)$ that consists of cycles defined as follows. Vertex set V contains four auxiliary vertices $v_1^0, v_2^0, v_3^0, v_4^0$ and four vertices $v_1^i, v_2^i, v_3^i, v_4^i$ per integer a_i, $i \in [n]$. Edge set E includes edges that connect each four vertices to a cycle by edges $e_j^i = \{v_j^i, v_{(j+1) \bmod 5}^i\}$ for $i \in \{0, \dots, n\}$ and $j \in [4]$. Furthermore, edge set E includes edges $\tilde{e}^i = \{v_1^i, v_2^i\}$ for all $i \in [n]$. The fixed edges of set E are defined by edges e_1^i for all $i \in [n]$ contained in set E^{fix}. The remaining edges are contained in set E^{free}. We set the capacity $u \equiv 1$ and define the weight c on all edges $e \in E$ by

$$c(e) = \begin{cases} 2w & \text{if } e = e_1^0, \\ 2a_i & \text{if } e = \tilde{e}^i,\ i \in [n], \\ a_i & \text{if } e = e_1^i,\ i \in [n], \\ 0 & \text{otherwise.} \end{cases}$$

Lastly, we define the degree restrictions $\boldsymbol{b} = (b^1, b^2)$ on all vertices $v \in V$ by

$$b^1(v) = \begin{cases} 1 & \text{for all } v \in \{v_1^i, v_2^i \mid i \in [n]\}, \\ 0 & \text{otherwise,} \end{cases} \qquad b^2(v) = \begin{cases} 1 & \text{if } v \in V \setminus \{v_3^0, v_4^0\}, \\ 0 & \text{otherwise.} \end{cases}$$

Overall, we obtain a feasible RobPM≡ instance $\widetilde{\mathcal{I}}$ constructed in polynomial time. Hence, it remains to show that $\mathcal{I}$ is a Yes-instance if and only if for instance $\widetilde{\mathcal{I}}$ a robust perfect $\boldsymbol{b}$-matching exists with weight of at most $\beta := 3w$.

Let A_1, A_2 be a feasible partition for instance $\mathcal{I}$. For the first scenario, we define matching f^1 by

$$f^1(e) = \begin{cases} 1 & \text{for } e = e_1^i \text{ if } a_i \in A_1, \\ 1 & \text{for } e = \tilde{e}^i \text{ if } a_i \in A_2, \\ 0 & \text{otherwise.} \end{cases}$$

The weight is

$$c(f^1) = \sum_{e \in E} c(e) f^1(e) \qquad = \sum_{i \in [n]:\ a_i \in A_1} c(e_1^i) f^1(e_1^i) + \sum_{i \in [n]:\ a_i \in A_2} c(\tilde{e}^i) f^1(\tilde{e}^i) = \sum_{a_i \in A_1} a_i + \sum_{a_i \in A_2} 2a_i = 3w.$$

For the second scenario, we define matching f^2 by

$$f^2(e) = \begin{cases} 1 & \text{for } e = e_1^0, \\ 1 & \text{for } e = e_1^i \text{ and } e = e_3^i \text{ if } a_i \in A_1, \\ 1 & \text{for } e = e_2^i \text{ and } e = e_4^i \text{ if } a_i \in A_2, \\ 0 & \text{otherwise.} \end{cases}$$

The weight is

$$\begin{aligned} c(f^2) &= \sum_{e \in E} c(e) f^2(e) \\ &= c(e_1^0) f^2(e_1^0) + \sum_{i \in [n]:\ a_i \in A_1} \big(c(e_1^i) f^2(e_1^i) + c(e_3^i) f^2(e_3^i)\big) + \sum_{i \in [n]:\ a_i \in A_2} \big(c(e_2^i) f^2(e_2^i) + c(e_4^i) f^2(e_4^i)\big) \\ &= 2w + \sum_{a_i \in A_1} (a_i + 0) + \sum_{a_i \in A_2} (0 + 0) = 3w. \end{aligned}$$

Consequently, we have constructed a robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f} = (f^1, f^2)$ with weight of $3w$.

Conversely, let $\boldsymbol{f} = (f^1, f^2)$ be a robust perfect $\boldsymbol{b}$-matching with weight $c(\boldsymbol{f}) = \max\{c(f^1), c(f^2)\} \leq 3w$. Let I_{fix} denote the set of indices that indicate which of the fixed edges $e_1^1, \dots e_1^n$ are matched in robust perfect $\boldsymbol{b}$-matching $\boldsymbol{f} = (f^1, f^2)$, i.e., $I_{\text{fix}} = \{i \in [n] \mid f^1(e_1^i) = f^2(e_1^i) = 1\}$. Perfect b^1-matching f^1 covers vertices v_1^i and v_2^i for all $i \in [n]$ by edge e_1^i or edge $\tilde{e}^i$. A lower bound on the weight is given by

$$\begin{aligned} c(f^1) &= \sum_{e \in E} c(e) f^1(e) \geq \sum_{i \in I_{\text{fix}}} c(e_1^i) + \sum_{i \in [n] \setminus I_{\text{fix}}} c(\tilde{e}^i) = \sum_{i \in I_{\text{fix}}} a_i + \sum_{i \in [n] \setminus I_{\text{fix}}} 2a_i = 2 \sum_{i=1}^{n} a_i - \sum_{i \in I_{\text{fix}}} a_i \\ &= 4w - \sum_{i \in I_{\text{fix}}} a_i. \end{aligned}$$

As $c(f^1) \leq 3w$ holds, we obtain $\sum_{i \in I_{\text{fix}}} a_i \geq w$. Perfect b^2-matching f^2 covers all vertices except v_3^0 and v_4^0. We note that the coverage of vertices v_1^0 and v_2^0 contributes $2w$ to the weight. A lower bound on the weight is given by

$$c(f^2) = \sum_{e \in E} c(e) f^2(e) \geq c(e_1^0) + \sum_{i \in I_{\text{fix}}} c(e_1^i) = 2w + \sum_{i \in I_{\text{fix}}} a_i.$$

As $c(f^2) \leq 3w$ holds, we obtain $\sum_{i \in I_{\text{fix}}} a_i \leq w$. Overall, we obtain $\sum_{i \in I_{\text{fix}}} a_i = w$. We define the sets $A_1 = \{a_i \mid i \in I_{\text{fix}}\}$ and $A_2 = \{a_i \mid i \in [n] \setminus I_{\text{fix}}\}$ which form a feasible partition.