

RESEARCH ARTICLE OPEN ACCESS

An Extended Formulation With Valid Inequalities for the Capacitated Steiner Arborescence Problem

Francesco Contu¹  | Massimo Di Francesco¹ | Enrico Gorgone¹ | Ishwar Murthy²

¹Department of Mathematics and Computer Science, University of Cagliari, Cagliari, Italy | ²Indian Institute of Management Bangalore, Bangalore, India

Correspondence: Francesco Contu (francesco.contu97@unica.it)

Received: 21 February 2025 | **Revised:** 24 April 2026 | **Accepted:** 26 May 2026

Keywords: integer programming | Steiner arborescence | valid inequalities

ABSTRACT

Given a directed graph, the Capacitated Steiner Arborescence Problem (CSAP) aims to determine the least-cost connection from the root node to terminal nodes requiring a demand through Steiner nodes coming with a capacity, such that there is a unique path from the root to each terminal. This paper presents a new extended formulation of the CSAP, which extends the classical multicommodity flow formulation of CSAP by introducing the notion of cardinality of terminals served by arcs. We show that our extended formulation is stronger than the classical one. We present two classes of inequalities and separation heuristics exploiting the cardinality effect of the extended formulation. The first class is a generalization of cardinality induced Cover and $(1, k)$ -Configuration inequalities. The second class is called the p -Arc Cardinality matching inequalities and is based on matching the number of commodities flowing on active arcs across a cut-set to its composite cardinality. The computational study clearly demonstrates the superiority of the extended formulation over the classical one. In particular, it consistently provides significantly tighter lower bounds at the root node. Moreover, for all hard instances, solving CSAP to optimality requires a number of branch-and-bound nodes that is several orders of magnitude larger under the classical formulation than under the extended one.

1 | Introduction

The Steiner Arborescence Problem (SAP) is a well-known problem in network design and has great relevance to applications in computer networks. Given a directed graph $G = (N, A)$, with $N = \{r\} \cup S \cup T$ and each arc $(i, j) \in A$ associated with a cost $c_{ij} \geq 0$, the Steiner Arborescence Problem is to connect the root node r to each terminal node $k \in T$ through Steiner nodes of set S , to obtain a least cost graph $G = (N', A')$, where $N' = \{r\} \cup S' \cup T$, $S' \subseteq S$ and $A' \subseteq A$. The graph $G = (N', A')$ has a tree structure, with a unique path from r to each $k \in T$. The importance of SAP derives from its close relationship to the more well-known Steiner tree problem defined on a graph $G = (N, E)$, where each $e \in E$ is an undirected link. The Steiner tree problem

has important applications in the design of logistics and telecommunication networks, as well as VLSI [1, 2]. It is well understood that, when costs $c_e \geq 0$ for each $e \in E$, the Steiner tree problem can be equivalently modeled as SAP by replacing each link $e = \{u, v\}$ by two directed arcs (u, v) and (v, u) [3]. Furthermore, as highlighted in [4, 5], SAP formulations provided tighter lower bounds from their LP relaxations than Steiner tree formulations with undirected links. Hence, our interest in SAP. In this paper, we consider the capacitated version of SAP, referred to as CSAP, in which each Steiner node $j \in S$ comes with a capacity $C_j > 0$, while each terminal node $k \in T$ requires a demand $d_k > 0$. As recognized in [2], ‘intermediate’ Steiner nodes operate with finite capacities in telecommunication networks, hence motivating the need to consider CSAP.

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2026 The Author(s). *Networks* published by Wiley Periodicals LLC.

Although there are many integer programming formulations of SAP in extant literature, they can all be broadly categorized into three, namely (1) *Cut*-based formulations, (2) *Tree*-based formulations and (3) *Multicommodity flow*-based formulations. Cut-based formulations are based on the observation that in any partition of $N = N_1 \cup N_2$, where $r \in N_1$ and $N_2 \cap T \neq \emptyset$, at least one arc connecting nodes in N_1 to nodes in N_2 must belong to A' . Tree-based formulations try to enforce a tree structure in every sub-graph defined by $N' \subseteq N$. In multicommodity flow-based formulations, a commodity is associated with each $k \in T$. The formulation here sends one unit of flow from r to each $k \in T$ of that commodity. Poltzin and Daneshmand [5] in their analysis suggest that existing extensions of the multicommodity flow formulations are superior to extensions of cut or tree-based formulations. The main thrust of this paper then is a new extended formulation of CSAP, which is based on further strengthening the classical multicommodity flow formulation.

1.1 | Classical Formulation of CSAP

Before presenting the classical multicommodity formulation, the following assumptions about CSAP are in order. First, the set of Steiner nodes S are treated as distinct from the terminal nodes T . However, if a terminal node $j \in T$ was also a candidate Steiner node with a capacity C_j , node j is split into two: $j_1 \in S$, with a capacity C_{j_1} , and $j_2 \in T$, with a demand d_{j_2} . Further, an arc (j_1, j_2) is created with cost equal to zero. The rest of the arcs entering and leaving j are now assigned to j_1 .

The cost c_{ij} associated with arc (i, j) , where $j \in S$ consists of two components. The first component is associated with establishing the link (i, j) . The other component is the cost of establishing the Steiner node $j \in S$. Such an assignment depends on the fact that at most one arc enters any node $j \in S$.

Finally, to ensure that the problem instance is not trivial, it is assumed that (i) $d_j \leq C_i$ for each $i \in S, j \in T$, and (ii) $\sum_{j \in T} d_j > C_i$ for each $i \in S$. The classical multicommodity flow formulation of CSAP uses two sets of binary variables: (a) $y_{ij} = 1$, if $(i, j) \in A'$, 0 otherwise, for each $(i, j) \in A$, and (b) $x_{ij}^k = 1$, if commodity k flows on arc $(i, j) \in A$, 0 otherwise.

The classical multicommodity flow formulation of CSAP is:

$$\begin{aligned}
 & (\mathbf{P}_{xy}) \\
 & \min \sum_{(i,j) \in A} c_{ij} y_{ij} \\
 & \text{s.t.} \\
 & \sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{ji}^k = \begin{cases} 1, i = r \\ -1, i = k \\ 0, i \neq r, k \end{cases} \quad \forall k \in T, \forall i \in N \quad (1) \\
 & x_{ij}^k \leq y_{ij} \quad \forall (i, j) \in A, \forall k \in T \quad (2) \\
 & \sum_{k \in T} d_k x_{ij}^k \leq C_j y_{ij} \quad \forall (i, j) \in \{A | j \in S\} \quad (3) \\
 & \sum_{i \in N} y_{ij} \leq 1 \quad \forall j \in S \quad (4a)
 \end{aligned}$$

$$\sum_{(i,j) \in A} y_{ij} \leq \sum_{(j,t) \in A} y_{jt} \quad \forall j \in S \quad (4b)$$

$$y_{ij}, x_{ij}^k \in \{0, 1\} \quad \forall (i, j) \in A, \forall k \in T \quad (5)$$

Constraints (1) in $(\mathbf{P}_{xy})$ enforce the flow conservation requirement wherein one unit of flow of commodity associated with $k \in T$ starts at node r and ends at $k \in T$. Constraints (2) ensure that the commodity associated with k does not flow on arc (i, j) , unless it is setup. Constraints (3) enforce the capacity restriction on each arc (i, j) where node $j \in S$, that is the flow of commodities on arc (i, j) cannot exceed C_j . Constraints (4a) enforce in part the tree structure wherein at most one arc enters each Steiner node $j \in S$. Finally, Constraints (4b), referred to as *node-balancing* constraints by Duin [6], enforce that the indegree of a Steiner node is less than or equal to its outdegree.

1.2 | Literature Review

This work addresses the capacitated version of the Steiner Arborescence Problem (CSAP), extending the uncapacitated formulation introduced by Wong et al. [3]. The underlying Steiner Arborescence Problem (SAP) is known to be NP-hard [7]. Despite its relevance, the capacitated variant has received limited attention in the literature. A closely related problem is the Capacitated Minimum Spanning Tree Problem (CMST), which has been extensively studied. However, CMST differs fundamentally from the setting considered here, as it assumes unit demands and defines capacity in terms of the maximum number of nodes in each subtree rooted at the depot.

In an early contribution, Gavish et al. [8] propose a single-commodity flow formulation of the problem, which enables a Dantzig–Wolfe decomposition and the development of a Lagrangian relaxation approach for the Capacitated Minimal Directed Tree Problem. The proposed relaxation is exploited to derive tight lower bounds on the optimal solution, as well as to design a heuristic for obtaining high-quality feasible solutions. The effectiveness of the procedure is demonstrated in computational tests. Later, a hop-index generalization of this formulation has been proposed by Gouveia and Martins [9]. Gouveia et al. [10] introduce a novel class of valid inequalities for the CMST problem with arbitrary node demands. These inequalities strengthen the classical multistar constraints. A Lagrangian relaxation dualizing these inequalities is proposed and tested. Computational experiment show that the new approach is particularly effective for random cost and sparse graphs. Yet, Gouveia et al. [11] introduce a flow-based formulation using binary variables z_{ij}^q for the CMST, which take the value 1 when a flow of q units pass through arc (i, j) . Unlike the previous formulations, which requires $O(n^2)$ constraints, this new model reduces the number of constraints to $O(n)$. Despite the simplification, its linear relaxation provides lower bounds that remain comparable to those of the original formulation.

The SAP was used as a powerful modeling tool for solving other network design problems (e.g., VLSI) [12–14]. As the SAP may provide a stronger LP relaxation than the STP, it offers the potential to support the development of more efficient solution algorithms. Furthermore, any instance of the Steiner Tree

Problem can be reformulated as a Steiner Arborescence Problem. Although extensive literature exists on the uncapacitated Steiner tree and its closely related SAP, there is little reported work on its capacitated versions. In fact, to the best of our knowledge, there is no reported work addressing capacitated versions as described above.

A recent survey on the Steiner Tree Problem, covering formulations and solution methods, is provided by Ljubić et al. [14]. The Steiner Tree Problem was explicitly formulated for the first time by Hakimi et al. [15], whereas Wong [3] is among the earliest contributions to introduce a multicommodity flow formulation for the Steiner tree problem on a directed graph. Specifically, given the same set of variables as in (P_{xy}) , the formulation in this paper consisted of constraints (1), (2) and (5). A dual ascent procedure was developed to solve the LP relaxation of the multicommodity flow formulation. Their formulation is valid for the SAP as well and can be easily adapted to the prize-collecting and other STP variants. Their computational results show that this formulation is capable of producing optimal or near-optimal solutions for the majority of the tested instances. Goemans and Myung [4] compared cut-based formulations to multicommodity flow formulations, both for Steiner tree and SAP. The classical cut-based formulation and multicommodity flow formulation are shown to be equivalent for SAP, which in turn are stronger than cut-based and multicommodity flow formulation for Steiner tree on undirected graphs. They introduced two extended formulations of a tree-based formulation by introducing binary variables y_j for each $j \in S$, besides link-based variables x_e for each $e \in E$. They showed the LP relaxation of their formulation to be as strong as multicommodity flow formulation. Liu [16] presented an extended formulation of SAP that is based on $|T| = 3$. Chopra et al. [17] used the cut-based formulation to develop a branch-and-cut procedure to solve the Steiner tree to optimality. Similarly, they report extensive computational results showing that optimal or near-optimal solutions are often obtained already at the root node of the branch-and-bound tree. Poltzin and Daneshmand [5] provide an excellent theoretical comparative analysis of the various formulations in terms of the strength of their LP relaxations. They showed that multicommodity flow-based formulations are at least as good as tree-based formulations in terms of lower bounds obtained from their respective LP relaxations. Furthermore, the ratio of their corresponding lower bounds can be arbitrarily large. In addition, they introduce a new formulation that extends the multicommodity flow formulation. Here, an additional set of flow variables that describes co-flow of two commodities, $k \in T$ and $l \in T$, along each arc $(i, j) \in A$ is introduced. Although the classical multicommodity flow formulation is described by $O(|A||T|)$ variables and constraints, the extended formulation of Poltzin and Daneshmand requires $O(|A||T|^2)$ variables and constraints. They demonstrate that this formulation is the strongest among all previously proposed formulations. In [18] Poltzin and Daneshmand approach the Steiner Problem in networks using an exact algorithm that integrates three main components: Relaxations, heuristics, and reductions. Relaxations are employed to derive lower bounds by relaxing feasibility constraints, thereby aiding in solving or approximating optimization problems. The authors proposed a family of new relaxations, demonstrated relationships between classical and new approaches, and constructed a hierarchy of relaxations. Heuristics provide “good” but not

necessarily optimal solutions, furnishing upper bounds crucial for exact algorithms and bound-based reductions. The authors introduced reduction-based heuristics that provide sharp upper bounds with high efficiency. Reductions simplify the size of the problem instances without compromising the optimal solution. The authors developed efficient realizations of classical tests and created a reduction package that performs stronger reductions than competing packages.

Another class of models that has been proposed is the family of hypergraphic formulations. These are based on the observation that the edge set of any Steiner tree can be uniquely partitioned into full Steiner components (FSCs) by splitting the tree at inner terminal nodes. In the context of the geometric Steiner tree problem, Warme [19] demonstrated that an optimal STP can be equivalently represented as a minimum-cost spanning tree in a specially constructed hypergraph whose nodes are terminals and whose hyperedges correspond to FSCs. The weight of each hyperedge equals the sum of the edge weights in the corresponding FSC. Although finding an MST in a hypergraph is NP-hard, hypergraphic Mixed Integer Programming (MIP) formulations can be derived from classical MIP models for spanning trees. For instance, the model using subtour elimination constraints (SECs) on undirected hypergraphs is equivalent to the model employing directed cuts on the corresponding directed hypergraph. Filipecki and Van Vyve [1] introduced MCF-based formulations employing aggregated-path variables. Their approach generalizes the classical MCF formulation by modeling flows over paths rather than solely over edges. For an integer parameter $\lambda \in 1, \dots, |V| - 1$, the formulation, denoted MCF- λ , considers MCFs from a root node to a terminal node by restricting attention to simple paths of length at most λ . In a recent work, Rehfeldt et al. [20] propose an exact approach for SPG by developing a novel implied bottleneck Steiner distance that estimates how critical an edge is in connecting terminals. If this distance is smaller than an edge's weight, that edge can be safely excluded, enabling stronger reduction rules than classical methods [21]. They also introduce conflict detection between edges using node and edge replacement ideas, which identifies edges that cannot coexist in a minimum Steiner tree. By combining the new distance measure with these conflict checks, they build a generalized extended reduction framework that systematically prunes the search space. The new methods were implemented within an enhanced branch-and-cut framework, with a strong emphasis on reduction techniques during presolving and domain propagation. Computational experiments demonstrate that their approach significantly outperforms the previous best state-of-the-art solver, achieving faster solution times across standard benchmarks, substantially reducing problem sizes during preprocessing (e.g., up to 75% for VLSI instances), and providing over sevenfold speedups on the hardest instances. Moreover, the solver solved 19 instances to optimality for the first time and efficiently handled large-scale Euclidean instances.

The work most closely related to the setting considered in this study is that of Bentz et al. [22]. They address a capacitated version in which each $j \in T$ can be thought of as having a unit demand, whereas each edge $e \in E$ comes with a capacity $c(e)$, defined as the number paths that it can handle from r to each $j \in T$. They studied various versions of this problem, obtained by classifying them along parameters, (i) fixed versus varying

size of $|T|$, (ii) edge length of zero versus not, and (iii) capacity $c(e) = 1$ versus greater than 1. By classifying the problem along these three dimensions, they were able to identify hard and easy versions in all but one. For those versions that were shown to be NP-Hard, they proposed polynomial bounded approximation algorithms with known worst case performance guarantees. For the same capacitated version addressed by Bentz et al. [22], Van den Eynde et al. [2] proposed a Capacitated Shortest path heuristic which they tested on a large set of problem instances. They were able to demonstrate their heuristic's ability to obtain good solutions for large size problems quickly. Di Puglia Pugliese et al. [23] considered a version of the problem in [22] in which all links $e \in E$ have a common demand K . They developed a Lagrangean relaxation-based procedure along with a primal heuristic from which they obtain a lower and upper bound on the optimal solution. They demonstrate the efficacy of their procedure on a large set of testing problems with very small gaps.

Several variants of the Steiner Tree Problems have been explored. Bentz et al. [24] study a robust variant of the capacitated Steiner tree and network design problem on directed graphs, motivated by applications such as wind farm cabling. Similarly to [22] each arc (i, j) that comes with a capacity u_{ij} . The authors address the challenge of ensuring connectivity from a root to multiple terminals under arc failures, whereas satisfying capacity constraints. They introduce several robustness models, including the possibility of protecting a subset of arcs, and focus on minimizing the cost of a resilient network that can sustain up to k arc failures without disconnecting any terminal. The analysis includes both tree and general network topologies, under the assumption of uniform demands. The authors first analyze a robust arborescence problem, proving its NP-completeness even under uniform unit demands, and propose multiple formulations balancing cost and robustness. For the general Capacitated Rooted k -Arc Connected Steiner Network problem, they introduce a bilevel optimization model along with equivalent cut-set and flow-based formulations and examine their theoretical properties. Further variants account for protected arcs, for which the formulations diverge. Two other variants of the Steiner Tree Problems are the Node-Weighted Steiner Tree Problem (NWSTP) and the Prize Collecting Steiner Tree Problem (PCSTP). Segev [25] introduced the NWSTP and proposed two Lagrangian relaxation bounding procedures for a special case where a single node is mandatory. These procedures are based on two mathematical models, the tree-type formulation and the flow formulation. Additionally, three simple heuristic methods were developed: One applying a subgradient method to the first Lagrangian relaxation, and two using greedy strategies. Both the heuristics and the bounding procedures were evaluated on complete graphs of up to 40 vertices. Bienstock et al. [26] were the first to propose a solution method for the PCSTP. They developed a heuristic inspired by Christofides's algorithm for the TSP adapted to the PCSTP yielding a heuristic with a worst-case performance ratio of 3. Leitner et al. [27] introduces a generalized Branch-and-Bound (B&B) framework designed for the asymmetric prize-collecting Steiner tree problem (APCSTP). They propose a B&B framework based on a dual ascent procedure. The proposed framework extends Wong's dual ascent algorithm by using a greedy scheme to iteratively adjust dual variables, increasing the dual objective while maintaining feasibility. The obtained lower bounds and dual information enable powerful

bound-based reduction tests. Extensive computational tests show that the framework is highly effective for various Steiner-type problems as it solves most benchmark instances—including very large ones—to optimality. The proposed method achieves speedups of up to hundred-fold compared to previous exact algorithms. Its success is due to tight lower and upper bounds, which enable strong graph reductions early in the computation. The framework also outperforms recent heuristics, but is less effective on highly symmetrical, reduction-resistant instances, where branch-and-cut methods remain better. Rehfeldt et al. [28] present exact algorithms for the PCSTP, combining new theoretical complexity insights with highly effective reduction techniques. First, they introduce a standard formulation (PrizeCut) for the general, unrooted PCSTP, which transforms the problem into a SAP instance by choosing an artificial root node. Then, they show that the PCSTP is Fixed-Parameter Tractable (FPT) when parameterized by the number of proper potential terminals, providing a strong theoretical basis for algorithms that explicitly distinguish this vertex class. Their main algorithmic contributions include the Prize-Constrained Distance (PCD), a novel walk-based distance measure that yields an edge elimination criterion provably stronger than existing bottleneck distance tests, leading to significant instance reductions. In addition, they introduce the Terminal-Regions Decomposition (TRD), a generalization of Voronoi decompositions that supports effective bound-based reductions during the pre-solving stage. Moreover, the left-rooted prize-constrained distance is employed to detect implicit roots, that is vertices that must belong to every optimal solution. This allows the PCSTP to be transformed into the Rooted PCSTP (RPCSTP). This rooting insight allowed them to introduce the PrizeRCut formulation, which uses a simpler transformation that avoids the artificial complexities and large constants necessary for the unrooted PrizeCut. As a result, the model yields a tighter Linear Programming relaxation and, consequently, stronger lower bounds for the optimal solution.

1.3 | Contributions of This Paper

Following the trend in the literature, we present an extended formulation of CSAP that introduces the notion of terminal cardinality on each arc $(i, j) \in \{A \mid j \in S\}$. By cardinality we mean the number of commodities—each associated with a terminal $k \in T$ that pass flow on arc (i, j) . Accordingly, the x and y variables in (P_{xy}) are disaggregated into at most $|T|$ sets. Although such a formulation is larger, consisting of at most $O(|A||T|^2)$ variables and constraints, we show that its LP relaxation provides a tighter lower bound than that provided by (P_{xy}) . We also present two classes of inequalities that exploit the notion of cardinality on arcs. The first ones are generalizations of the Cover and $(1-k)$ Configuration inequalities. Given the knapsack-type constraints (3) on each arc, the Cover and $(1-k)$ Configuration inequalities are strengthened based on the cardinality of commodities that are active on that arc. The second class of inequalities, called p -Arc Cardinality matching inequalities, is derived from matching the number of commodities flowing on p active arcs across a cut-set to the composite cardinality of these p arcs. It needs to be mentioned here that the notion of cardinalities to disaggregate variables has been successfully tried and implemented earlier on the Generalized Assignment Problem, as is reported in [29], as well as the Concentrator Location Problem, reported in [30]. In both these

papers, the use of cardinality induced Cover and (1- k) Configuration inequalities have been successfully implemented as well. What is indeed new in this paper, is the introduction of the p -Arc Cardinality matching inequalities.

Finally, an implementation of a branch-and-cut procedure using IBM ILOG CPLEX Optimizer software [31] is reported, wherein at the root node, the LP relaxation of the extended formulation is solved, followed by identifying and adding the proposed inequalities when violated, together with the internal cuts generated by CPLEX. The performance of our branch-and-cut procedure is compared to that of the formulation in (P_{xy}) in CPLEX on both benchmark and synthetic test problems. The results demonstrate that the lower bounds obtained at the root node of the branch-and-bound tree from our branch-and-cut procedure are superior to that obtained from the classical formulation (P_{xy}) . More significantly, for those problem instances where the duality gap obtained at the root node of the branch-and-bound tree (P_{xy}) was on the higher side, the number of branch-and-bound nodes needed to verify optimality was found to be orders of magnitude higher as opposed to the extended formulation.

The rest of this paper is organized as follows. In Section 2, the extended formulation of CSAP is presented. We show that the LP relaxation of the extended formulation is higher than that obtained from the LP relaxation of (P_{xy}) . In Section 3, we present (a) cardinality induced Cover inequalities, (b) cardinality induced (1, k)-Configuration inequalities, and (c) p -Arc Cardinality matching inequalities. We also describe separation heuristics for each type of inequality. In Section 4, the results of a detailed computational study are reported to assess the effectiveness of our approach in terms of strength of the bounds, the number of branch-and-bound nodes required, and time required to solve the instances to optimality. Concluding remarks are presented in Section 5.

2 | Extended Formulation of CSAP

Associated with each Steiner node $i \in S$, let S_i denote the maximum number of commodities that can pass through it. To determine S_i , the demands d_j for each $j \in T$ is first sorted in non-increasing order. Next, for each $i \in S$, items in T are 'packed in a knapsack' with capacity C_i in the same order, till its capacity is about to be exceeded. Each $i \in S$ is now split into $i(1), \dots, i(S_i)$ nodes, resulting in an extended graph $G_e = (N_e, A_e)$, with $N_e = \{r\} \cup S_e \cup T$. A node $i(s) \in S_e$ represents the Steiner node $i \in S$ with exactly s commodities entering it, if selected. The cost associated with arcs entering node $j(s) \in S_e$ is the same as the arcs entering node $j \in S$, that is $c_{ij(s)} = c_{ij}$. To keep subsequent exposition simple, nodes in S_e will generally be referred to as i or j instead of $i(s)$ or $j(s)$, unless specific reference needs to be made to s . Finally, the arc set $A'_e \subset A_e$ is induced by the node set $\{r\} \cup S_e$. Given this disaggregation, the variable set y is defined exactly as in (P_{xy}) , whereas variable set z replaces x . The extended formulation is:

$$\begin{aligned} & (P_{zy}) \\ & \min \sum_{(i,j) \in A_e} c_{ij} y_{ij} \\ & s.t. \end{aligned}$$

$$\sum_{j \in N_e} z_{ij}^k - \sum_{j \in N_e} z_{ji}^k = \begin{cases} 1, i = r, \\ -1, i = k \\ 0, i \neq r, k \end{cases} \quad \forall k \in T, \forall i \in N_e \quad (6)$$

$$\begin{aligned} z_{ij}^k & \leq y_{ij} & \forall (i, j) \in A_e \\ & & \forall k \in T \end{aligned} \quad (7)$$

$$\begin{aligned} \sum_{k \in T} d_k z_{ij(s)}^k & \leq C_j y_{ij(s)} & \forall (i, j(s)) \in A'_e \\ & & \forall s = 1, \dots, S_j \end{aligned} \quad (8)$$

$$\begin{aligned} \sum_{k \in T} z_{ij(s)}^k & = s y_{ij(s)} & \forall (i, j(s)) \in A'_e \\ & & \forall s = 1, \dots, S_j \end{aligned} \quad (9)$$

$$\sum_{i \in N_e} \sum_{s=1}^{S_j} y_{ij(s)} \leq 1 \quad \forall j \in S \quad (10)$$

$$\begin{aligned} z_{ij}^k, y_{ij} & \in \{0, 1\} & \forall (i, j) \in A_e \\ & & \forall k \in T \end{aligned} \quad (11)$$

In (P_{zy}) , (6) describes the flow conservation constraints for each commodity $k \in T$ starting from node r to its destination $k \in T$. Constraints (7), known as *variable upper bound* (VUB) constraints ensure that no flow of commodity k occurs along arc (i, j) if it is not setup. Constraints (8) represents knapsack type constraints, defined for each $(i, j(s)) \in A'_e$ and cardinality s . *Cardinality* constraints (9) enforce the requirement that if arc $(i, j(s))$ is part the Arborescence tree, then exactly s commodities must flow on it. Finally, constraints (10) ensure the tree structure wherein at most one arc enters each $j \in S$. A straightforward strengthening of (P_{zy}) is based on the observation that the number of commodities entering a node $j \in S$ either remains the same on a singleton arc leaving it or gets split up into flows of two or more arcs. Consequently, for any arc $(i(s_1), j(s_2)) \in A'_e$ to be part of the solution, $s_1 \geq s_2$. Therefore, all arcs $(i(s_1), j(s_2)) \in A'_e$ where $s_1 < s_2$ can be eliminated from $G_e = (N_e, A_e)$. The following example illustrates in part the construction of $G_e = (N_e, A_e)$ and the associated formulation (P_{zy}) from $G = (N, A)$.

Example 1. Consider the graph $G = (N, A)$, with $N = \{r\} \cup S \cup T$ and A consists of arcs that originate from r to each node in S and T . Furthermore, A also consists of arcs originating from each node in S to remaining nodes in S and each node in T . Here, $S = \{1, 2\}$ with capacities $C_1 = 10$ and $C_2 = 11$, respectively, whereas $T = \{3, 4, 5, 6\}$, with demands, $d_3 = 3$, $d_4 = 4$, $d_5 = 5$ and $d_6 = 6$. Consequently, $S_1 = S_2 = 2$. The resulting arc set $A'_e = \{(r, 1(1)), (r, 1(2)), (r, 2(1)), (r, 2(2)), (1(1), 2(1)), (2(1), 1(1)), (1(2), 2(1)), (1(2), 2(2)), (2(2), 1(1)), (2(2), 1(2))\}$. The remaining arcs in A_e connect nodes in $\{r\} \cup S_e$ to nodes in T . The figure below illustrates the node set $\{r\} \cup S_e$ and the arc set A'_e generated for the above example (Figure 1).

Note that as mentioned earlier, there are no arcs $(1(1), 2(2))$ and $(2(1), 1(2))$ in A'_e . Note as well that constraint (10) on node $2(2)$ is $y_{r,2(2)} + y_{1(2),2(2)} \leq 1$.

It is worth noting that (P_{zy}) and (P_{xy}) are not strictly equivalent. Although every feasible solution in (P_{zy}) translates to an

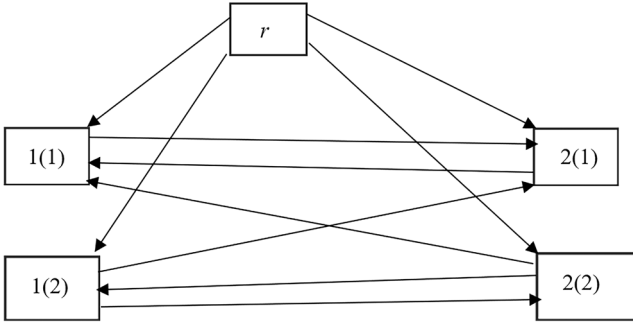


FIGURE 1 | Illustration of node set $\{r\} \cup S_e$ and the arc set A'_e in Example (1).

equivalent solution in (P_{xy}) with the same objective function value, the converse is not true. Specifically, in (P_{xy}) , for some $(i, j) \in A$, $y_{ij} = 1$, but $z_{ij}^k = 0$ for each $k \in T$ is feasible. There is no such equivalent solution in (P_{zy}) . However, in all known applications, $c_{ij} \geq 0$ for all $(i, j) \in A$. Therefore, in any optimal solution to (P_{xy}) , if $x_{ij}^k = 0$ for each $k \in T$ for some $(i, j) \in A$, then $y_{ij} = 0$. The same then holds for (P_{zy}) .

Consider the following LP polyhedral:

$$LP(x, y) = \{(x, y) \in R^{|A||T|+|A|} \mid (1) \dots (4), x \geq 0, y \geq 0\} \quad (12)$$

$$LP(z, y) = \{(z, y) \in R^{|A_c||T|^2+|A_c||T|} \mid (6) \dots (10), z \geq 0, y \geq 0\} \quad (13)$$

For some $(z, y) \in LP(z, y)$, consider the following aggregation,

$$x_{ij}^k = \sum_{i(s=1)}^{i(S_i)} \sum_{j(s=1)}^{j(S_j)} z_{i(s)j(s)}^k \quad (14)$$

$$y_{ij} = \sum_{i(s=1)}^{i(S_i)} \sum_{j(s=1)}^{j(S_j)} y_{i(s)j(s)} \quad (15)$$

Let $LP_a(x, y)$ denote a projection of each $(z, y) \in LP(z, y)$ onto the x - y space, defined as:

$$LP_a(x, y) = \{(x, y) \in LP(x, y) \mid (14), (15), \forall (z, y) \in LP(z, y)\} \quad (16)$$

To clarify, (16) represents a transformation mapping every $(z, y) \in LP(z, y)$ to the corresponding $(x, y) \in LP_a(x, y)$.

Lemma 1. $LP_a(x, y) \subseteq LP(x, y)$.

Proof: Consider a $(z, y) \in LP(z, y)$, which satisfies (6–8) and (10). By applying (14) and (15) to (z, y) , $(x, y) \in LP_a(x, y)$ is obtained. In the process, (6–8) and (10) get aggregated to become (14), respectively. Hence, $(x, y) \in LP(x, y)$.

To show that $LP_a(x, y) \subseteq LP(x, y)$ it suffices to identify an instance of $(x, y) \in LP(x, y)$ for which there is no equivalent $(z, y) \in LP(z, y)$. That is, there is no disaggregation of (x, y) possible, which results in a $(z, y) \in LP(z, y)$ with the same objective function value. Consider a $(\hat{x}, \hat{y}) \in LP(x, y)$ in which on an arc $(i, j) \in A$ the commodity set $K \subset T$ has positive flows on it. The commodity set K is partitioned into K_1 and K_2 , with $|K_1| \geq 1$

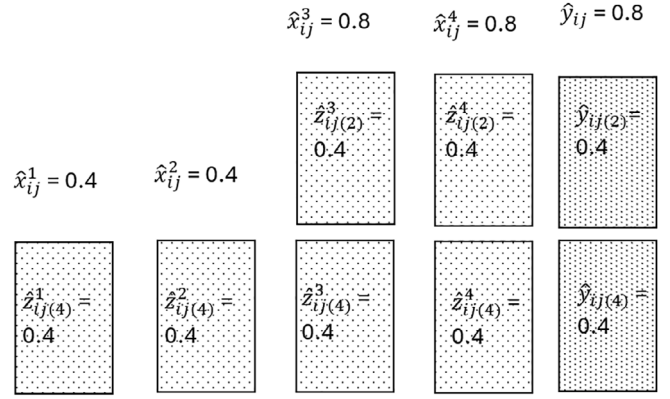


FIGURE 2 | Illustration of Lemma 1 using data in Example (2).

and $|K_2| \geq 1$. For each $k \in K_1$, $\hat{x}_{ij}^k = x_1$, each $k \in K_2$, $\hat{x}_{ij}^k = x_2$, with $0 < x_1 < x_2 = \hat{y}_{ij}$. Furthermore, the knapsack constraint (3) is satisfied exactly. Thus,

$$\sum_{k \in K_1} d_k x_1 + \sum_{k \in K_2} d_k x_2 = C_j \hat{y}_{ij} \quad (17)$$

The only way to disaggregate $(\hat{x}, \hat{y})$ on (i, j) to obtain (z, y) that satisfies cardinality constraint (9) is to first split the node j into $j(s_1)$ and $j(s_2)$, with $s_1 = |K_1| + |K_2|$ and $s_2 = |K_2|$. Furthermore, (i) $y_{ij(s_1)} = x_1$, (ii) $y_{ij(s_2)} = x_2 - x_1$, (iii) $z_{ij(s_1)}^k = x_1$ for each $k \in K$, and (iv) $z_{ij(s_2)}^k = x_2 - x_1$ for each $k \in K_2$, would satisfy (7) and (9). Since $\hat{y}_{ij} = x_2$, it follows from (17) that $C_j > \sum_{k \in K_2} d_k$. Therefore, $C_j y_{ij(s_2)} > \sum_{k \in K_2} d_k z_{ij(s_2)}^k$. It then follows that $\sum_{k \in K} d_k z_{ij(s_1)}^k > C_j y_{ij(s_1)}$ for (17) to be satisfied. This clearly violates knapsack constraint (8). Hence, $LP_a(x, y) \subseteq LP(x, y)$.

To illustrate Lemma 1 consider the following example:

Example 2. Consider the arc $(i, j) \in A$ where $j \in S$ with associated $C_j = 14$. Let the fractional solution on (i, j) be as follows: $\hat{y}_{ij} = 0.8$, $\hat{x}_{ij}^1 = 0.4$, $\hat{x}_{ij}^2 = 0.4$, $\hat{x}_{ij}^3 = 0.8$, $\hat{x}_{ij}^4 = 0.8$. The demands of the four commodities are, $d_1 = d_2 = 8$, whereas $d_3 = d_4 = 3$. Note that the fractional solution $(\hat{x}, \hat{y})$ in $LP(x, y)$ satisfies (2) and the knapsack constraint (3) exactly. The only way the fractional solution $(\hat{x}, \hat{y})$ can be disaggregated to obtain $(\hat{z}, \hat{y})$ is to set: (a) $K_1 = \{1, 2\}$, $K_2 = \{3, 4\}$, (b) $s_1 = 4$, $s_2 = 2$, and (c) $\hat{y}_{ij(4)} = \frac{4}{10}$, $\hat{y}_{ij(2)} = \frac{4}{10}$, $\hat{z}_{ij(4)}^1 = \frac{4}{10}$, $\hat{z}_{ij(4)}^2 = \frac{4}{10}$, $\hat{z}_{ij(4)}^3 = \frac{4}{10}$, $\hat{z}_{ij(4)}^4 = \frac{4}{10}$, $\hat{z}_{ij(2)}^3 = \frac{4}{10}$, $\hat{z}_{ij(2)}^4 = \frac{4}{10}$. Note that $(\hat{z}, \hat{y})$ satisfies (7) and (9). Observe that while $3\hat{z}_{ij(2)}^3 + 3\hat{z}_{ij(2)}^4 < 14\hat{y}_{ij(2)}$, $8\hat{z}_{ij(4)}^1 + 8\hat{z}_{ij(4)}^2 + 3\hat{z}_{ij(4)}^3 + 3\hat{z}_{ij(4)}^4 > 14\hat{y}_{ij(4)}$. This is amply illustrated in Figure 2 below.

Besides Lemma 1, another important point to note about the strength of the formulation in (P_{zy}) is that it subsumes the flow-balance constraints of Duin [6]. The flow-balance constraints have been successfully used to strengthen both cut-based and multicommodity formulations of the Steiner Tree and Steiner Arborescence problems. Let $\delta^-(j(s))$ denote the set of all arcs entering $j(s)$, and $\delta^+(j(s))$ the set of all arcs leaving $j(s)$. For (P_{zy}) , the flow-balance constraints, the equivalent of (4b), are defined for each $j(s) \in S$ as

$$\sum_{(i,j(s)) \in \delta^-(j(s))} y_{ij(s)} \leq \sum_{(j(s),i) \in \delta^+(j(s))} y_{j(s)i} \quad (18)$$

Constraint (18) essentially describes in part the tree structure wherein the sum of y variables associated with arcs entering $j(s)$ is less than or equal to the sum of y variables leaving $j(s)$. The following proposition shows that due to the elimination of all arcs $(i(s_1), j(s_2)) \in A'_e$ where $s_1 < s_2$ from $G_e = (N_e, A_e)$ and constraints (10), every $(z, y) \in LP(z, y)$ satisfies (18), thus obviating the need to incorporate them in (P_{zy}) .

Lemma 2. Every $(z, y) \in LP(z, y)$ satisfies (18) for each $j(s) \in S$.

Proof: Consider a $(z, y) \in LP(z, y)$ in which a $j(s) \in S$ is 'active' with a positive flow of commodities entering it, that is $\sum_{(i,j(s)) \in \delta^-(j(s))} y_{ij(s)} > 0$. Let $U \subseteq T$ denote the set of commodities with positive flows in one or more arcs $(i, j(s)) \in \delta^-(j(s))$, that is for each $k \in U$, $z_{ij(s)}^k > 0$ for some $(i, j(s)) \in \delta^-(j(s))$. It follows from cardinality constraint (9) that

$$\sum_{(i,j(s)) \in \delta^-(j(s))} \sum_{k \in U} z_{ij(s)}^k = s. \quad (19a)$$

We denote the end-node of each $(j(s), i) \in \delta^+(j(s))$ as having cardinalities of $s_1, \dots, s_p$ respectively. Since all arcs $(i(s_a), j(s_b)) \in A'_e$ are such $s_a \geq s_b$, it follows that $s_l \leq s$, for each $l = 1, \dots, p$. Due to flow conservation constraints (6), it follows that

$$\sum_{(i,j(s)) \in \delta^-(j(s))} \sum_{k \in U} z_{ij(s)}^k = \sum_{(j(s),i) \in \delta^+(j(s))} \sum_{k \in U} z_{j(s)i}^k. \quad (19b)$$

From the satisfaction of cardinality constraints (9) on each $(j(s), i(s_l)) \in \delta^+(j(s))$ it follows that

$$s * \left(\sum_{(i,j(s)) \in \delta^-(j(s))} y_{ij(s)} \right) = \sum_{(j(s),i(s_l)) \in \delta^+(j(s))} s_l * y_{j(s)i(s_l)}. \quad (19c)$$

Equation (19c) can be rewritten as follows:

$$\left(\sum_{(i,j(s)) \in \delta^-(j(s))} y_{ij(s)} \right) = \sum_{(j(s),i(s_l)) \in \delta^+(j(s))} \frac{s_l}{s} y_{j(s)i(s_l)}. \quad (19d)$$

One possibility is that $s_l = s$, for each $l = 1, \dots, p$. In this case, it follows that (18) is satisfied exactly. The other possibility is that $s_l < s$ for one or more arcs $(j(s), i(s_l)) \in \delta^+(j(s))$. Since $\left(\frac{s_l}{s}\right) \leq 1$ for each $l = 1, \dots, p$, with one or more l where $\left(\frac{s_l}{s}\right) < 1$, it follows that in (18) the left-hand-side is less than the right-hand-side.

3 | Valid Inequalities for (P_{zy})

In this section, we present three classes of inequalities, (i) cardinality induced Cover, (ii) cardinality induced $(1, k)$ -Configuration, which we will refer to as $(1, \hat{p}_k)$ -Configuration inequalities, and (iii) p -Arc Cardinality Matching inequalities. In addition, separation heuristics are devised for each of the considered inequalities.

3.1 | Cardinality Induced Cover Inequality

We now present a generalization of the well-known Cover inequality, which is derived from knapsack constraint (8) on arc $(i, j) \in A'_e$. Let

$$H(x, y) = \text{Conv} \left\{ (x, y) \in B^{|A||T|+|A|} \mid (1) \dots (5) \right\}, \quad (20)$$

$$H(z, y) = \text{Conv} \left\{ (z, y) \in B^{|A_e||T|^2+|A_e||T|} \mid (6) \dots (11) \right\}. \quad (21)$$

where $B = \{0, 1\}$. The classical cover inequality that is valid for $H(x, y)$, defined on an arc $(i, j) \in A$ is as follows. Let $V \subseteq T$ with $|V| = v$, satisfy conditions:

- i) $\sum_{k \in U} d_k \leq C_j, \forall U \subset V, |U| = u \geq 1$
- ii) $\sum_{k \in U^+} d_k > C_j, \forall U^+ \subseteq V, |U^+| = u + 1$

Given the above conditions, the (v, u) -cover inequality is valid for $H(x, y)$, that is the convex hull of feasible solutions obtained from (P_{xy}) :

$$\sum_{k \in V} x_{ij}^k \leq uy_{ij} \quad (22)$$

Definition 1. Let $L \subseteq T$ with $0 \leq k \leq |L|$. Then, $D^*(L, k) = \min \left\{ \sum_{j \in L} d_j x_j \mid \sum_{j \in L} x_j = k, x_j \in \{0, 1\}^{|L|} \right\}$.

Given a set of L commodities, $D^*(L, k)$ represents a selection of k items with minimum cumulative demand. Now consider an arc $(i, j(s)) \in A'$, where $s \geq 2$. Given that the set $V \subseteq T$ satisfies conditions (i) and (ii) above, let $\hat{R}_s \subset V$ with $|\hat{R}_s| = \hat{r}_s < \min\{v, s\}$ in turn satisfy conditions,

$$\sum_{j \in \hat{R}_s} d_j + D^*(T - \hat{R}_s, s - \hat{r}_s) \leq C_j, \quad (23)$$

$$\sum_{j \in \hat{R}_{s+1}} d_j + D^*(T - \hat{R}_{s+1}, s - \hat{r}_s - 1) > C_j, \forall \hat{R}_{s+1} \subseteq V, |\hat{R}_{s+1}| = \hat{r}_s + 1 \quad (24)$$

Then the following $(v, s, \hat{r}_s)$ -cover inequality is valid for $H(z, y)$, the convex hull of feasible solutions in (P_{zy}) :

$$\sum_{k \in V} z_{ij(s)}^k \leq \hat{r}_s y_{ij(s)}. \quad (25)$$

We now show that the $(v, s, \hat{r}_s)$ -cover inequalities added to $LP(z, y)$ results in 'deeper' cuts than the addition of (v, u) -cover inequality to $LP(x, y)$. Note that, for a given $V \subseteq T$, since $D^*(T - \hat{R}_{s+1}, s - \hat{r}_s - 1) \geq 0$, it follows that $\hat{r}_s \leq u$, for each $s \geq 2$. For $s = 1$, $\hat{r}_s = 1$. Furthermore, for a given V , while just one (v, u) -cover inequality can be constructed for $H(x, y)$, several $(v, s, \hat{r}_s)$ -cover inequalities can be constructed for $H(z, y)$, one for each $s \geq 2$. That being the case, if for a given V , constraints in (25) are aggregated over $i(s)$ and $j(s)$, then from (14), we obtain $\sum_{k \in V} x_{ij}^k \leq \sum_{i(s=1)}^{j(S_i)} \sum_{j(s=1)}^{j(S_j)} \hat{r}_s y_{i(s)j(s)}$. Due to (10), $\sum_{i(s=1)}^{j(S_i)} \sum_{j(s=1)}^{j(S_j)} y_{i(s)j(s)} \leq 1$. With $\hat{r}_s \leq u$, for each $s \geq 2$ and for $s = 1$, $\hat{r}_s = 1$, it follows that $\sum_{i(s=1)}^{j(S_i)} \sum_{j(s=1)}^{j(S_j)} \hat{r}_s y_{i(s)j(s)} \leq uy_{ij}$. Therefore, the collection of $(v, s, \hat{r}_s)$ -cover inequalities derived for a given V , dominates the corresponding (v, u) -cover inequality. The following example illustrates the strength of $(v, s, \hat{r}_s)$ -cover inequalities vis-à-vis (v, u) -cover inequalities.

Example 3. Consider an instance in which $|T| = 8$, with demands d_j in sorted order from largest to smallest being,

[14, 14, 14, 14, 14, 7, 7, 7]. Consider the arc (r, j) , with $C_j = 50$ and $V = \{1, \dots, 5\}$. Then a $(5, 3)$ -cover inequality obtained for $H(x, y)$ is $\sum_{k \in V} x_{rj}^k \leq 3y_{rj}$. What the $(5, 3)$ -cover inequality says is that at most three from the first five commodities in T can be accommodated on arc (r, j) , since the total demand of any four commodities exceeds the capacity of 50. Note that since $S_j = 5$, in $G_e = (N_e, A_e)$ node j is split into $\{j(1), \dots, j(5)\}$. Then, on arc $(r, j(5))$, for every $\hat{R}_s \subset V$ with $\hat{r}_s = 2$, Conditions (23) and (24) are satisfied, since $D^*(T - \hat{R}_s, 5 - \hat{r}_s) = 21$, while $D^*(T - \hat{R}_{s+1}, 5 - \hat{r}_s - 1) = 14$. Note that $D^*(T - \hat{R}_s, 5 - \hat{r}_s) = 21$ implies that 3 commodities with the smallest demands from the set $(T - \hat{R}_s)$ adds up to 21. However, two more commodities are needed from $\hat{R}_s$ on $(r, j(5))$ to enable a cardinality of 5. The additional demand being 28, the capacity constraint is satisfied. However, with $D^*(T - \hat{R}_{s+1}, 5 - \hat{r}_s - 1) = 14$, three more commodities are needed from $\hat{R}_s$ on $(r, j(5))$, whose additional demand is 42, resulting in the capacity constraint being violated. Hence, the $(v, s, \hat{r}_s)$ -cover inequality derived is $\sum_{k \in V} z_{rj(5)}^k \leq 2y_{rj(5)}$. On arc $(r, j(4))$, with $\hat{r}_s = 3$, $D^*(T - \hat{R}_s, 4 - \hat{r}_s) = 7$, whereas $D^*(T - \hat{R}_{s+1}, 4 - \hat{r}_s - 1) = 0$. Therefore, here $\sum_{k \in V} z_{rj(4)}^k \leq 3y_{rj(4)}$. On arcs $(r, j(3))$, $(r, j(2))$ and $(r, j(1))$, the $\hat{r}_s$ values implied are 3, 2 and 1, respectively, which are rendered redundant by constraints (9).

We now identify a partial solution $(z, y) \in LP(z, y)$ that violates the $(v, s, \hat{r}_s)$ -cover inequalities above. Consider the partial solution, $y_{rj(5)} = \frac{1}{2}$, $z_{rj(5)}^4 = z_{rj(5)}^5 = z_{rj(5)}^7 = z_{rj(5)}^8 = \frac{1}{2}$, $z_{rj(5)}^2 = \frac{1}{14}$, $z_{rj(5)}^6 = \frac{6}{14}$, $y_{rj(4)} = z_{rj(4)}^4 = z_{rj(4)}^5 = z_{rj(4)}^6 = z_{rj(4)}^7 = \frac{1}{2}$. It is easy to see that this solution satisfies VUB constraints (7) and cardinality constraints (8) on both arcs, $(r, j(5))$ and $(r, j(4))$. The knapsack constraint (8) on arc $(r, j(5))$ is satisfied exactly with both the left-hand-side and right-hand-side equal to 25. The knapsack constraint on arc $(r, j(4))$ is also satisfied with its left-hand-side and right-hand-side equal to 21 and 25, respectively. However, the $(v, s, \hat{r}_s)$ -cover inequality on $(r, j(5))$ is violated with its l.h.s equal to $\frac{15}{14}$ and its r.h.s equal to 1. Upon aggregating (14) and (15), the equivalent $(x, y) \in LP_d(x, y)$ obtained is: $y_{rj} = 1$, $x_{rj}^4 = x_{rj}^5 = x_{rj}^7 = 1$, $x_{rj}^8 = \frac{1}{2}$, $x_{rj}^2 = \frac{1}{14}$, $x_{rj}^6 = \frac{13}{14}$. This solution satisfies the (v, u) -cover inequality, $\sum_{k \in V} x_{rj}^k \leq 3y_{rj}$, with its l.h.s equal to $\frac{29}{14}$ while its r.h.s equal to 3. This illustrates the fact that the $(v, s, \hat{r}_s)$ -cover inequalities dominate the (v, u) -cover inequalities.

We now describe a simple separation heuristic for identifying a $(v, s, \hat{r}_s)$ -cover inequality that violates a fractional solution $(z, y) \in LP(z, y)$. We first identify an arc $(i, j(s)) \in A'_e$ with $y_{ij(s)} > 0$, and a set $Co_{ij(s)} \subseteq T$ with positive flows along $(i, j(s))$, that is $z_{ij(s)}^k > 0$, for each $k \in Co_{ij(s)}$. Furthermore, (z, y) satisfies (8) on $(i, j(s))$ exactly. The items in $Co_{ij(s)}$ are now sorted from smallest to largest using function, $g(y_{ij(s)}, z_{ij(s)}^k, d_j) = (y_{ij(s)} - z_{ij(s)}^k + 0.1)/d_j$. This function value is used to select $V \subseteq Co_{ij(s)}$, with those $k \in Co_{ij(s)}$ with smaller value, more preferred. The identification of $V \subseteq Co_{ij(s)}$ starts with $v = 2$. In the first iteration, V comprises of the first v items in $Co_{ij(s)}$. If (23) or (24) are not satisfied with $\hat{r}_s = v - 1$, then in the next iteration, V comprises of items 2 to $v + 1$ and tested for (23) and (24). Thus, in iteration l , items l to $v + l - 1$ are tested for (23) and (24) till $v + l - 1$ equals $|Co_{ij(s)}|$. At this point v is incremented by 1 and the search continues

ALGORITHM 1 | Algorithm_ $\hat{r}_s$.

1. **for each** $(i, j(s)) \in A'_e$ **such that** $y_{ij(s)} > 0$:
2. $Co_{ij(s)} = \{k \in T \mid z_{ij(s)}^k > 0\}$
5. **sort** elements in $Co_{ij(s)}$ in terms of $(y_{ij(s)} - z_{ij(s)}^k + 0.1)/d_k$
6. $v = 2$
8. **while** $v < |Co_{ij(s)}|$:
9. $l = 2$
10. **while** $v + l - 1 < |Co_{ij(s)}|$:
11. construct the set V considering the first l elements of $Co_{ij(s)}$
13. if (23) and (24) are satisfied:
14. add the inequality (25) corresponding to V with $\hat{r}_s = v - 1$
15. $l = l + 1$
16. $v = v + 1$

till $v = |Co_{ij(s)}|$. Whenever, both (23) and (24) are satisfied for a given V , then the cover inequality (25) with $\hat{r}_s = v - 1$ is set up. This inequality is further lifted by expanding V to selectively include items in $T - V$ in the following way. First, all items $k \in T - V$ whose $d_k \geq \max\{d_k \mid k \in V\}$ are included. Next, for those $k \in T - V$ whose $d_k < \max\{d_k \mid k \in V\}$ if $D^*(V, \hat{r}_s) + d_k + D^*(T - V, s - \hat{r}_s + 1) > C_j$, then k is included in V . After this lifting procedure, if the resulting cover inequality (25) is violated, it is added as a cut (Algorithms 1 and 2).

3.2 | Cardinality Induced (1, p)-Configuration Inequality

Analogously to Cover inequality, the $(1, k)$ -Configuration inequality [32, 33] is derived from knapsack constraints. In this paper, since 'k' is often used to refer to cardinality, in the interest of clarity in exposition, the $(1, k)$ -Configuration inequality is henceforth referred to as $(1, p)$ -Configuration inequality. The basic idea behind the $(1, p)$ -configuration inequality is as follows. Given a 'knapsack' with a capacity of C , let $V \subset T$ denote the set of commodities that can be accommodated in the knapsack. There exists a 'big' item $q \in T - V$, which cannot be accommodated along with all items in V . More specifically, only all sets $P_{-1} \subset V$, consisting of $p - 1$ items can be accommodated, along with item q . Formally, on some arc $(i, j) \in A_e$, let $V \subset T$, with $|V| = v$ and $q \in T - V$, let

$$\sum_{j \in V} d_j \leq C, \quad (26)$$

$$\sum_{j \in P \cup \{q\}} d_j > C, \forall P \subseteq V, |P| = p, 2 \leq p \leq v \quad (27)$$

$$\sum_{j \in P_{-1} \cup \{q\}} d_j \leq C, \forall P_{-1} \subset V, |P_{-1}| = p - 1 \quad (28)$$

For some $R_v \subseteq V$, with $|R_v| = r_v$ satisfying $p \leq r_v \leq v$, the $(1, p)$ -configuration inequalities that are valid for $H(x, y)$ is

$$(r_v - p + 1)x_{ij}^q + \sum_{k \in R_v} x_{ij}^k \leq r_v y_{ij} \quad (29)$$

Observe that for a given V , (29) represents large number of inequalities, one for each selection of R_v out of V . For the same V , the $(1, p)$ -configuration inequalities can be strengthened to be valid for $H(z, y)$ as follows. On an arc $(i, j(s)) \in A'_e$, consider a $\hat{P}_v \subseteq V \subset T$ with $|\hat{P}_v| = \hat{p}_v \leq s$ and a $q \in T - V$, that satisfies

$$\sum_{k \in \hat{P}_v} d_k + D^*(T - \hat{P}_v, s - \hat{p}_v) \leq C_j, \quad (30)$$

$$\sum_{k \in \hat{P}_v \cup \{q\}} d_k + D^*(T - \hat{P}_v - q, s - \hat{p}_v - 1) > C_j, \forall \hat{P}_v \subseteq V, |\hat{P}_v| = \hat{p}_v \quad (31)$$

$$\sum_{k \in \{\hat{P}_{v-1} \cup q\}} d_k + D^*(T - \hat{P}_{v-1} - q, s - \hat{p}_v) \leq C_j, \forall \hat{P}_{v-1} \subset V, |\hat{P}_{v-1}| = \hat{p}_v - 1 \quad (32)$$

Note that arc $(i, j(s))$ requires that s commodities flow on it, if chosen. Therefore, condition (30) states that it is possible to accommodate all commodities in $\hat{P}_v$, as well as $s - \hat{p}_v$ commodities chosen from $T - \hat{P}_v$ on arc $(i, j(s))$. On the other hand, condition (31) states that if commodity $q \in T - \hat{P}_v$ were to flow on $(i, j(s))$, then it is not possible to accommodate any $\hat{P}_v \subseteq V$, with $|\hat{P}_v| = \hat{p}_v$, even if the smallest set of $s - \hat{p}_v - 1$ commodities are chosen from the set $T - \hat{P}_v - \{q\}$. However, as per condition (32), in the presence of commodity q , it is possible to accommodate any $\hat{P}_{v-1} \subset V$, with $|\hat{P}_{v-1}| = \hat{p}_v - 1$, along with the remaining $s - \hat{p}_v$ commodities from $T - \hat{P}_{v-1} - \{q\}$. Upon satisfaction of conditions (30–32), for some $R_v \subseteq V$, with $R_v = r_v$ where $\hat{p}_v \leq r_v \leq v$, the following set of $(1, \hat{p}_v)$ -configuration inequalities that is valid for $H(z, y)$:

$$(r_v - \hat{p}_v + 1)z_{ij(s)}^q + \sum_{k \in R_v} z_{ij(s)}^k \leq r_v y_{ij(s)} \quad (33)$$

It is worth noting that for a given $R_v \subseteq V$, one $(1, p)$ -configuration inequality (29) is constructed, while several $(1, \hat{p}_v)$ -configuration inequalities (33) are obtained, one for each $s \geq r_v$. For the case, $s = r_v$, $D^*(T - \hat{P}_v - q, s - \hat{p}_v - 1) = 0$ in (31). Consequently, $\hat{p}_v = p$, and the effect of inequality (33) is the same as inequality (29). However, for each $s > r_v$, $D^*(T - \hat{P}_v - q, s - \hat{p}_v - 1) > 0$, and consequently $\hat{p}_v < p$, in each such case. For all $s < r_v$, it suffices to set $\hat{p}_v = p$ in (33), and the resulting inequality will be valid for $H(z, y)$. That is because, when $s = r_v$, the left-hand-side of (33) is at most r_v . Therefore, when $s < r_v$, due to (9) and (33) is naturally satisfied. The following example illustrates the relative strength of $(1, \hat{p}_v)$ -configuration inequalities vis-à-vis the $(1, p)$ -configuration inequality.

Example 4. Consider the following instance in which $|T| = 11$, with demands d_j in sorted order being, [22, 15, 15, 15, 14, 14, 14, 6, 6, 6, 6]. Let $V = \{5, 6, 7\}$, and a Steiner node $j \in S$ with a capacity $C_j = 50$. Then, defining $R_v = V$, $q = 1$ and $p = 3$ on arc (r, j) , a $(1, p)$ -configuration inequality that can be constructed is: $x_{rj}^1 + x_{rj}^5 + x_{rj}^6 + x_{rj}^7 \leq 3y_{rj}$. Note that in $G_e = (N_e, A_e)$, since $S_j = 5$, node j gets split into $\{j(1), \dots, j(5)\}$. On arc $(r, j(4)) \in A_e$, that is $s = 4$, let $\hat{P}_v \subset R_v$ with $\hat{p}_v = 2$. Here, $D^*(T - \hat{P}_v, s - \hat{p}_v) = D^*(T - \hat{P}_v, 4 - 2) = 12$

represents the sum of the two smallest demands from the set [22, 15, 15, 15, 14, 6, 6, 6, 6]. The sum of demands of every set of two commodities $\hat{P}_v \subset R_v$ is 28. Therefore, (28) is satisfied for every $\hat{P}_v \subset R_v$. In (31), commodity $q = 1$ is added to $\hat{P}_v$, resulting in the total demand of these three commodities is $28 + 22 = 50$. To account for the cardinality of $s = 4$, the smallest demand of $D^*(T - \hat{P}_v - q, s - \hat{p}_v - 1) = 6$. The left-hand-side in (31) being $56 > 50$, it is satisfied. In (32), the left-hand-side is the sum of the demands one commodity in $\hat{P}_{v-1}$, that of $q = 1$, and two smallest commodities from the remaining, which is $D^*(T - \hat{P}_{v-1} - q, s - \hat{p}_v) = 12$. Hence, the left-hand-side in (32) is $22 + 14 + 12 = 48$, and it is satisfied. Consequently, the $(1, \hat{p}_v)$ -configuration inequality obtained on arc $(r, j(4))$ is, $2z_{rj(4)}^q + z_{rj(4)}^5 + z_{rj(4)}^6 + z_{rj(4)}^7 \leq 3y_{rj(4)}$. On arc $(r, j(5))$, since $s = 5$, at most two commodities from the set V can be accommodated, noting that the demands of the two commodities add up to 28, whereas the minimum total from the rest is $3 * 6 = 18$. Therefore, $r_v = 2$. Note that if $q = 1$ is to be accommodated on arc $(r, j(5))$, then no commodity from V can be accommodated. Demand $d_1 = 22$, whereas any commodity in V has a demand of 14. Three more commodities from the rest have to be accommodated, the minimum total demand being $3 * 6 = 18$. Thus, the total demand adds up to 54, exceeding the capacity of 50. However, commodity $q = 1$ can be feasibly accommodated along with commodities 8–11. Therefore, the $(1, \hat{p}_v)$ -configuration inequality obtained is, $2z_{rj(5)}^q + z_{rj(5)}^5 + z_{rj(5)}^6 + z_{rj(5)}^7 \leq 2y_{rj(5)}$.

Consider the following partial solution, $y_{rj(4)} = \frac{1}{2}$, $z_{rj(4)}^q = z_{rj(4)}^5 = z_{rj(4)}^8 = \frac{1}{2}$, $z_{rj(4)}^9 = z_{rj(4)}^{10} = \frac{1}{4}$, $y_{rj(5)} = \frac{1}{2}$, $z_{rj(5)}^q = z_{rj(5)}^8 = z_{rj(5)}^9 = z_{rj(5)}^{10} = \frac{1}{2}$, $z_{rj(5)}^5 = z_{rj(5)}^{11} = \frac{1}{4}$. Observe that this solution satisfies constraints (7–9), on both arcs $(r, j(4))$ and $(r, j(5))$. Although the $(1, \hat{p}_v)$ -configuration inequality associated with arc $(r, j(4))$ is satisfied exactly, the one associated with arc $(r, j(5))$ is violated with the left-hand-side equal to $\frac{5}{4}$ and the right-hand-side equal to 1. By aggregating the (z, y) solution above using (14) and (15), the (x, y) solution obtained is, $y_{rj} = 1$, $x_{rj}^q = x_{rj}^8 = 1$, $x_{rj}^5 = x_{rj}^9 = x_{rj}^{10} = \frac{3}{4}$, $x_{rj}^{11} = \frac{1}{4}$. This solution satisfies (2) and (3) and the $(1, p)$ -configuration inequality.

A brief description of a separation heuristic that identifies a violated $(1, \hat{p}_v)$ -configuration inequality now follows (Algorithm 2). An arc $(i, j(s)) \in A'_e$ with $s \geq 2$ and associated $y_{ij(s)} > 0$, along with the set $Co_{ij(s)} \subseteq T$ with positive flows on $(i, j(s))$ is identified. The terminal q is identified as $d_q = \max \{d_j | j \in Co_{ij(s)}\}$, whereas $V = Co_{ij(s)} - q$. Next, as with the heuristic for identifying a $(v, s, \hat{r}_s)$ -cover inequality, V is sorted in increasing order, in terms of $g(y_{ij(s)}, z_{ij(s)}^k, d_j) = (y_{ij(s)} - z_{ij(s)}^k + 0.1)/d_j$. To obtain $\hat{p}_v$, first the smallest subset $R_v \subset V$ with $|R_v| = r_v$ that satisfy $\sum_{j \in R_v} d_j + D^*(T - R_v - q, s - r_v - 1) > C_j - d_q$ is obtained. Next, set $\hat{P}_v = R_v$ and $\hat{p}_v = r_v$. Suppose that an item $j^* \in V - R_v$ satisfies, (i) $D^*(R_v, \hat{p}_v - 1) + d_{j^*} + D^*(T - \hat{P}_{v-1} - j^* - q, s - \hat{p}_v - 1) > C_j - d_q$ where $\hat{P}_{v-1} \subset \hat{P}_v$ and $|\hat{P}_{v-1}| = \hat{p}_v - 1$, and (ii) $\sum_{j \in R_v} d_j + d_{j^*} + D^*(T - R_v - j^*, s - r_v - 1) \leq C_j$. Whenever this happens, $R_v = R_v \cup j^*$ and $r_v = r_v + 1$. This process of expanding R_v continues till all items in $V - R_v$ have been explored or that $r_v = s$.

ALGORITHM 2 | Algorithm_{π₁}.

```

1. for each  $(i, j(s)) \in A'_e$  such that  $y_{ij(s)} > 0$  :
2.    $Co_{ij(s)} = \left\{ k \in T \mid z_{ij(s)}^k > 0 \right\}$ 
3.    $q = \operatorname{argmax}_{k \in Co_{ij(s)}} d_k$ 
4.    $V = Co_{ij(s)} - q$ 
5.   sort elements in  $V$  in terms of  $(y_{ij(s)} - z_{ij(s)}^k + 0.1)/d_k$ 
6.   find  $R_v \subset V$  such that  $\sum_{j \in R_v} d_j + D^*(T - R_v - q, s - |R_v| - 1) > C_j - d_q$ 
7.    $\hat{P}_v = R_v$  and  $r_v = \hat{P}_v = |R_v|$ 
8.   while  $r_v < s$  and  $V - R_v \neq \emptyset$ :
9.     find a  $j^* \in V - R_v$  such that:
10.     $D^*(R_v, \hat{P}_v - 1) + d_{j^*} + D^*(T - \hat{P}_{v-1} - j^* - q, s - \hat{P}_v - 1) > C_j - d_q$ 
11.     $\sum_{j \in R_v} d_j + d_{j^*} + D^*(T - R_v - j^*, s - r_v - 1) \leq C_j$ 
12.    else if such  $j^*$  not found:
13.      stop
14.    if (33) is satisfied:
15.      add the inequality (33) corresponding to  $R_v, r_v$  and  $\hat{P}_v$ 
16.       $R_v = R_v \cup j^*$ 
17.       $r_v = r_v + 1$ 

```

3.2.1 | P-Arc Cardinality Matching Inequalities

The p -Arc Cardinality matching inequality is defined over $p \geq 2$ arcs that form a cut-set separating the set of terminals in $U \subseteq T$ from root node r . That is, $N_e = r \cup S_e \cup T$ is partitioned into two dichotomous sets, $N_e(1) = \{r\} \cup S_e(1) \cup T - U$ and $N_e(2) = S_e(2) \cup U$. Let $A_c = \{(i, j) \in A_e \mid i \in N_e(1), j \in N_e(2)\}$ and $\hat{A}_p \subseteq A_c$ be such that $|\hat{A}_p| = p$. The selection of set $S_e(2)$ is minimal, in the sense that every $j(s) \in S_e(2)$, is the end-node of an arc $(i, j(s)) \in \hat{A}_p$. The essential idea behind the p -Arc Cardinality matching inequalities is that in any feasible solution to (P_{zy}) , $|U| = u$ commodities will flow on all arcs $\hat{a} \in \hat{A}_p$. Note as well that, due to (10), a cardinality (or number of commodities) of s is associated with an arc $a \in A_a$ if $a = (i, j(s))$. This includes those arcs $a = (i, j(1))$, where $j(1) \in T$. Given this, the cardinalities associated with all arcs $\hat{a} \in \hat{A}_p$ must add up to u . We now present two variants of the p -Arc Cardinality matching inequalities that reflect this idea.

3.2.2 | P-Arc Cardinality Matching Inequality-I

We begin with the simplest one. Consider the solution $(z, y) \in LP(z, y)$ in which $z_{ij}^k = 0$ for each $k \in U$ and $(i, j) \in A_e - \hat{A}_p$. As well, $z_{ij}^k = 0$ for each $k \in T - U$ and $(i, j) \in \hat{A}_p$. Thus, there is a clear partition of flows, in which all commodities $k \in U$ flow entirely on arcs in $\hat{A}_p$, while all commodities $k \in T - U$ flow entirely on some arcs in $A_e - \hat{A}_p$. This implies that $\sum_{(i, j(s)) \in \hat{A}_p} s y_{ij(s)} = u$. That being the case, it follows that

$$\sum_{(i, j(s)) \in \hat{A}_p} \sum_{k \in U} z_{ij(s)}^k \leq \sum_{(i, j(s)) \in \hat{A}_p} s y_{ij(s)}, \quad (34)$$

is valid for $H(z, y)$.

ALGORITHM 3 | Algorithm_{π₁}.

```

1.  $s = s_{min}$ 
2.  $COUNT = 0$ 
3.  $\pi_1 = 0$ 
4. while  $s \leq s_{max}$ :
5.   if  $(u - COUNT) < s \mid J_{node}(s) \mid$ :
6.      $\pi_1 = \pi_1 + \lfloor (u - COUNT)/s \rfloor$ 
7.      $s_f = s$ 
8.     stop
9.   else:
10.     $\pi_1 = \pi_1 + \lfloor J_{node}(s) \rfloor$ 
11.     $COUNT = COUNT + s * \lfloor J_{node}(s) \rfloor$ 
12.     $s = s + 1$ 

```

Now consider the following optimization problem:

$$\pi_1 = \max \left\{ \sum_{(i, j(s)) \in \hat{A}_p} y_{ij(s)} \mid (36a), (36b), (36c) \right\} \quad (35)$$

where

$$\sum_{(i, j(s)) \in \hat{A}_p} s y_{ij(s)} = u \quad (36a)$$

$$\sum_{(i, j(s)) \in \hat{A}_p} y_{ij(s)} \leq 1 \forall j(s) \in N_e(2) \quad (36b)$$

$$y_{ij(s)} \in \{0, 1\} \forall (i, j(s)) \in \hat{A}_p \quad (36c)$$

It is clear from (35) that $0 \leq \pi_1 - \sum_{(i, j(s)) \in \hat{A}_p} y_{ij(s)}$, provided (35a–35c) are satisfied. The following inequality,

$$\sum_{(i, j(s)) \in \hat{A}_p} \sum_{k \in U} z_{ij(s)}^k \leq \sum_{(i, j(s)) \in \hat{A}_p} (s - 1) y_{ij(s)} + \pi_1 \quad (37)$$

is obtained as the sum of (34) and $0 \leq \pi_1 - \sum_{(i, j(s)) \in \hat{A}_p} y_{ij(s)}$. Since (34) is valid for $H(z, y)$, so is inequality (37), which we will refer to as p -Arc Cardinality Matching inequality I.

To obtain π_1 in (37), the optimization problem in (35) is solved by viewing it as a single-machine job shop problem. Here, each arc $(i, j(s)) \in \hat{A}_p$ is a 'job' requiring s units time to process and all jobs $(i, j(s)) \in \hat{A}_p$, have a common due date of u . The objective then is to maximize throughput of jobs. Hence, the solution to (33) uses the Shortest-Processing-Time (SPT) rule in which jobs requiring less processing time are processed before those requiring more time. To implement this rule, the following lists are used: (i) $Card_Node(j(s)) = s$ for each $j(s) \in N_e(2)$, and (ii) $J_{node}(s) = \{j(s) \in N_e(2) \mid Card_Node(j(s)) = s\}$. That is, list $Card_Node(j(s))$ assigns a cardinality of s to each node $j(s) \in N_e(2)$, while list $J_{node}(s)$ denotes the set of all nodes $j(s) \in N_e(2)$ whose cardinality is s . As well, (i) $s_{min} = \min \{Card_Node(j(s)) \mid j(s) \in N_e(2)\}$, (ii) $s_{max} = \max \{Card_Node(j(s)) \mid j(s) \in N_e(2)\}$. The following algorithm is formally stated to solve (35):

In Algorithm_{π₁}, essentially nodes (or jobs) $j(s) \in N_e(2)$ associated with the smallest cardinality (or processing time) are selected before moving on to nodes (jobs) with higher cardinality

(processing time). Note that while the inequality in (37) is defined in terms of selection of arcs in $\hat{A}_p$, in *Algorithm_ π_1* (Algorithm 3), π_1 is determined as the maximum number of nodes selected in $N_e(2)$. This is due to (36b) which enforces that at most one arc in $\hat{A}_p$ enters each node in $N_e(2)$. The following example illustrates an instance where a fractional solution $(z, y) \in LP(z, y)$ violates (37).

Example 5. Consider a $(z, y) \in LP(z, y)$ in which $\hat{A}_p = \{(r, 1(3)), (r, 2(3)), (r, 3(3)), (r, 4(3)), (r, 5(2)), (r, 6(2))\}$, $u = 12$ with $U = \{10, \dots, 21\}$, and the corresponding partial solution is, $y_{r1(3)} = y_{r2(3)} = y_{r3(3)} = y_{r4(3)} = \frac{5}{6}$, $y_{r5(2)} = y_{r6(2)} = \frac{1}{2}$. Note that by inspection, $\sum_{(i,j(s)) \in \hat{A}_p} \sum_{k \in U} z_{ij(s)}^k = \sum_{(i,j(s)) \in \hat{A}_p} s y_{ij(s)} = 12$. In this example, $s_{\min} = 2$, $s_{\max} = 3$, $J_{\text{node}}(2) = \{5(2), 6(2)\}$ and $J_{\text{node}}(3) = \{1(3), 2(3), 3(3), 4(3)\}$. In the first iteration of *Algorithm_ π_1* (Algorithm 3) with $s = 2$, $(u - \text{COUNT}) = 12 \geq s \mid J_{\text{node}}(s) \mid = 2 * 2 = 4$. Therefore, $\pi_1 = 2$ and $\text{COUNT} = 4$. In the second iteration, $(u - \text{COUNT}) = 8 < s \mid J_{\text{node}}(s) \mid = 3 * 4 = 12$. Therefore, $\pi_1 = \pi_1 + \lfloor (u - \text{COUNT})/s \rfloor = 2 + \lfloor (12 - 4)/3 \rfloor = 4$. The procedure stops with $\pi_1 = 4$. This partial solution violates the resulting inequality (37) since its left-hand-side is 12 and its right-hand-side is $1 + \frac{20}{3} + 4 = \frac{35}{3}$. This is illustrated in Figure 3 below.

It is worth observing that in Example (5) above, the p -Arc Cardinality Matching inequality in (37) is constructed with all commodities in U flowing exclusively on arcs in $\hat{A}_p$. However, in some instances it helps to remove some of those arcs from $\hat{A}_p$ where the flow amount is very small. Consequently, the total flow in $\hat{A}_p$ is less than u . However, by removing these arcs, the π_1 value obtained from solving (33) is reduced much more, thereby resulting in a violation. This is illustrated in the following example.

Example 6. Consider the following fractional solution $(z, y) \in LP(z, y)$ in which $\hat{A}_p = \{(r, 1(2)), (r, 2(2)), (r, 3(2)), (r, 4(3)), (r, 5(3)), (r, 6(3)), (r, 7(3))\}$, $U = \{11, \dots, 16\}$ and therefore $u = 6$. The fractional y values are: $y_{r1(2)} = \frac{1}{6}$, $y_{r2(2)} = \frac{3}{4}$, $y_{r3(2)} = \frac{1}{12}$, $y_{r4(3)} = \frac{1}{12}$, $y_{r5(3)} = \frac{1}{6}$, $y_{r6(3)} = \frac{1}{12}$, $y_{r7(3)} = 1$. Observe that $\sum_{(i,j(s)) \in \hat{A}_p} s y_{ij(s)} = 6$. Since

every commodity in U is allowed to flow on every arc in $\hat{A}_p$, to construct the p -Arc Cardinality Matching inequality I, the actual flow of commodities, that is the z values do not matter. Upon executing *Algorithm_ π_1* (Algorithm 3), $\pi_1 = 3$ is obtained. Consequently, the right-hand-side of (37) is $\left(\frac{1}{6} + \frac{3}{4} + \frac{1}{12}\right) + 2 * \left(\frac{1}{12} + \frac{1}{6} + \frac{1}{12} + 1\right) + 3 = \frac{20}{3}$, whereas the left-hand-side value of (37) is 6. Hence, the inequality (37) so constructed is not violated. However, if $(r, 3(2))$ is removed from $\hat{A}_p$, then upon executing *Algorithm_ π_1* (Algorithm 3), $\pi_1 = 2$ is obtained. Therefore, the right-hand-side of (37) now becomes $\left(\frac{1}{6} + \frac{3}{4}\right) + 2 * \left(\frac{1}{12} + \frac{1}{6} + \frac{1}{12} + 1\right) + 2 = \frac{67}{12}$, while the left-hand-side reduces to $\frac{70}{12}$. This result is a violation.

3.2.3 | P-Arc Cardinality Matching Inequality-II

It is clear from (37) that in the construction of p -Arc Cardinality Matching Inequality I, all commodities in U are allowed to flow on all arcs in $\hat{A}_p$. In the next version of the p -Arc Cardinality Matching inequality, the flow of each commodity $k \in U$ on arcs $(i, j(s)) \in \hat{A}_p$ is restricted in the following way. Based on a current fractional solution $(z, y) \in LP(z, y)$, define $\text{Com_Card}(\hat{s}) = \{k \in U \mid z_{ij(s)}^k > 0 \text{ for at least one } (i, j(s)) \in \hat{A}_p\}$ for each $s_{\min} \leq \hat{s} \leq s_{\max}$. The set $\text{Com_Card}(\hat{s})$, denotes the collection of all commodities with positive flows on at least one arc in $\hat{A}_p$, whose end-node has a cardinality of $\hat{s}$. Essentially then, only those commodities $k \in \text{Com_Card}(\hat{s})$, are allowed to flow on arcs $(i, j(s)) \in \hat{A}_p$. Based on this notion, the following optimization problem is solved to obtain π_2 .

$$\pi_2 = \max \left\{ \sum_{(i,j(s)) \in \hat{A}_p} y_{ij(s)} \mid (36a), (36b), (36c), (39a), (39b), (39c), (39d) \right\} \quad (38)$$

where

$$\sum_{(i,j(s)) \in \hat{A}_p} z_{ij(s)}^k \leq 1 \quad \forall k \in U, \quad (39a)$$

$$z_{ij(s)}^k \leq y_{ij(s)}, \forall k \in \text{Com_Card}(\hat{s}), \forall (i, j(s)) \in \hat{A}_p, \quad (39b)$$

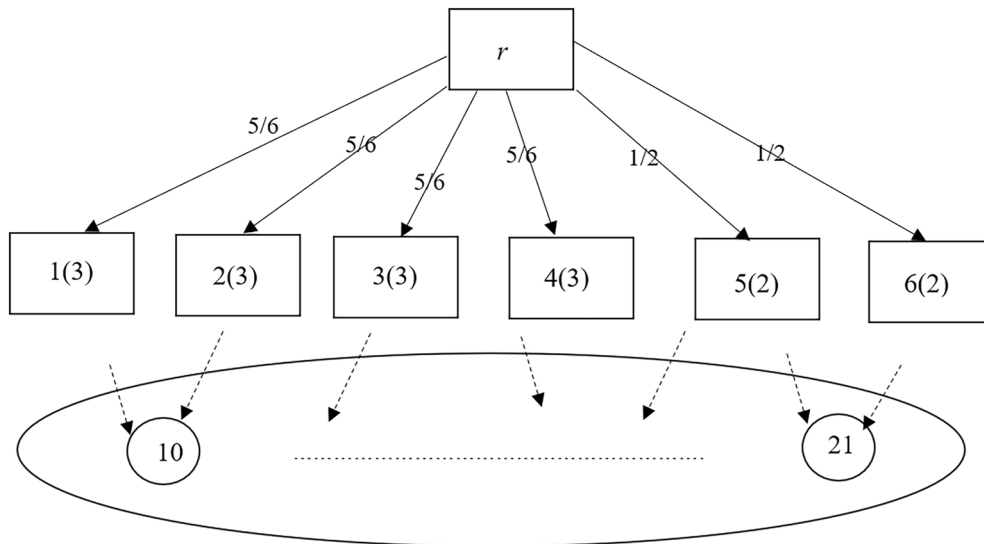


FIGURE 3 | Illustration of Example (5).

$$\sum_{k \in \text{Com_Card}(\hat{s})} z_{ij(\hat{s})}^k \leq \hat{s} y_{ij(\hat{s})}, \forall (i, j(\hat{s})) \in \hat{A}_p, \quad (39c)$$

$$z_{ij(\hat{s})}^k \in \{0, 1\}, \forall k \in \text{Com_Card}(\hat{s}), (i, j(\hat{s})) \in \hat{A}_p. \quad (39d)$$

Upon obtaining π_2 , the resulting inequality obtained is:

$$\sum_{(i, j(s)) \in \hat{A}_p} \sum_{k \in \text{Com_Card}(\hat{s})} z_{ij(s)}^k \leq \sum_{(i, j(s)) \in \hat{A}_p} (s-1) y_{ij(s)} + \pi_2 \quad (40)$$

Note that (40) is the sum of

$$\sum_{(i, j(s)) \in \hat{A}_p} \sum_{k \in \text{Com_Card}(\hat{s})} z_{ij(s)}^k \leq \sum_{(i, j(s)) \in \hat{A}_p} s y_{ij(s)} \quad (41)$$

and $0 \leq \pi_2 - \sum_{(i, j(s)) \in \hat{A}_p} y_{ij(s)}$. Note that in (41), only those commodities belonging to $\text{Com_Card}(s)$, are allowed to flow on each arc $(i, j(s)) \in \hat{A}_p$. In contrast, in (34), all commodities $k \in U$, are allowed to flow on all arcs $(i, j(s)) \in \hat{A}_p$. Since (34) is valid for $H(z, y)$, so is (41). Furthermore, the solution to (38) ensures that $\pi_2 - \sum_{(i, j(s)) \in \hat{A}_p} y_{ij(s)} \geq 0$. Therefore, (40) is a valid inequality for $H(z, y)$.

Note that due to restrictions on the flow of commodities on arcs in $\hat{A}_p$, as well as constraint (36a), the problem in (38) could be infeasible. In which case, $u = u - 1$ in (36a) and (38) is resolved. This is done iteratively till a feasible solution is found. The algorithm employed to solve the problem in (38) can be viewed as maximizing the number of bins needed to pack items in U . Each arc $(i, j(\hat{s})) \in \hat{A}_p$ is a 'bin' with a capacity of $\hat{s}$, whereas each item in U occupies a unitary volume. To facilitate this algorithm, the list $J_{\text{arc}}(\hat{s}) = \{(i, j(\hat{s})) \in \hat{A}_p | j(\hat{s}) \in J_{\text{node}}(\hat{s})\}$ is used. All items $k \in \text{Com_Card}(\hat{s})$ are allowed to be packed in arcs $(i, j(\hat{s})) \in J_{\text{arc}}(\hat{s})$, that is arcs whose ending node has a cardinality of $\hat{s}$.

Our proposed algorithm first packs arcs having lower capacity with available commodities before moving to arcs with higher capacity. Hence, the proposed procedure begins with $\hat{s} = s_{\min}$, terminating at $\hat{s} = s_{\max}$. For arcs with a capacity of $\hat{s}$, the number of arcs needed to accommodate maximum possible commodities in $\text{Com_Card}(\hat{s})$ is obtained as, $\alpha(\hat{s}) = \min \{ \lfloor |\text{Com_Card}(\hat{s})| / \hat{s} \rfloor, |J_{\text{arc}}(\hat{s})| \}$. The number of commodities in $\text{Com_Card}(\hat{s})$ that are actually accommodated in arcs $(i, j(\hat{s}))$, is $\hat{s} * \alpha(\hat{s}) \leq |\text{Com_Card}(\hat{s})|$. The process of selecting the $\hat{s} * \alpha(\hat{s})$ number of commodities for $\text{Com_Card}(\hat{s})$ and assigning them to arcs in $J_{\text{arc}}(\hat{s})$ is as follows. Define $\text{Count}_{\hat{s}^+}(k) = \sum_{s=\hat{s}+1}^{s_{\max}} |k \cap \text{Com_Card}(s)|$, for each $k \in \text{Com_Card}(\hat{s})$. It denotes the number of cardinality levels above $\hat{s}$ in which commodity $k \in \text{Com_Card}(\hat{s})$ has at least one arc with positive flow on it. The commodities in $\text{Com_Card}(\hat{s})$ are then sorted in non-decreasing order, based on which the first $\hat{s} * \alpha(\hat{s})$ number of commodities are assigned to arcs in $J_{\text{arc}}(\hat{s})$. These $\hat{s} * \alpha(\hat{s})$ commodities are removed from $\text{Com_Card}(s)$, for all $s = \hat{s} + 1, \dots, s_{\max}$. The idea behind this step is to ensure first those commodities that are allowed only on arcs in $J_{\text{arc}}(\hat{s})$ are selected. A formal statement of this algorithm is:

The following example illustrates the construction of inequality (40) using *Algorithm* _{π_2} (*Algorithm* 4).

ALGORITHM 4 | *Algorithm* _{π_2} .

1. $\pi_2 = 0$
2. $\hat{s} = s_{\min}$
4. **while** $\hat{s} \leq s_{\max}$:
5. **if** $\text{Com_Card}(\hat{s}) \neq \emptyset$:
6. $\alpha(\hat{s}) = \min \{ \lfloor |\text{Com_Card}(\hat{s})| / \hat{s} \rfloor, |J_{\text{arc}}(\hat{s})| \}$
7. $\pi_2 = \pi_2 + \alpha(\hat{s})$
8. **For each** $k \in \text{Com_Card}(\hat{s})$:
9. $\text{Count}_{\hat{s}^+}(k) = \sum_{s=\hat{s}+1}^{s_{\max}} |k \cap \text{Com_Card}(s)|$
10. **Select** the first $\hat{s} * \alpha(\hat{s})$ elements of $\text{Com_Card}(\hat{s})$ in non-decreasing order of $\text{Count}_{\hat{s}^+}(k)$
11. **Delete** these $\hat{s} * \alpha(\hat{s})$ elements from $\text{Com_Card}(s)$ for all $s = \hat{s} + 1, \dots, s_{\max}$
12. $s = s + 1$

Example 7. Consider the following partial fractional solution, (z, y) : $y_{r1(3)} = y_{r2(3)} = \frac{2}{3}$, $y_{r3(2)} = \frac{1}{2}$, $y_{r4(2)} = \frac{1}{4}$, $y_{r2(2)} = y_{r5(2)} = \frac{1}{8}$. Here, $U = \{1, 2, \dots, 6\}$ and hence $u = 6$. The associated flow variables of the commodities in U are as follows: $z_{r1(3)}^1 = z_{r2(3)}^1 = z_{r3(2)}^1 = \frac{1}{3}$, $z_{r1(3)}^2 = \frac{1}{3}$, $z_{r2(3)}^2 = \frac{2}{3}$, $z_{r1(3)}^3 = \frac{2}{3}$, $z_{r2(3)}^3 = \frac{1}{3}$, $z_{r1(3)}^4 = \frac{1}{3}$, $z_{r2(3)}^4 = \frac{1}{2}$, $z_{r3(2)}^4 = \frac{1}{6}$, $z_{r1(3)}^5 = \frac{1}{3}$, $z_{r2(3)}^5 = \frac{1}{6}$, $z_{r4(2)}^5 = \frac{1}{4}$, $z_{r2(2)}^5 = \frac{1}{8}$, $z_{r3(2)}^6 = \frac{1}{2}$, $z_{r4(2)}^6 = \frac{1}{4}$, $z_{r2(2)}^6 = z_{r5(2)}^6 = \frac{1}{8}$. This fractional solution is illustrated in Figure 4 below. Note that Figure 4 illustrates the flows from each node $j(s) \in N_e(2)$ to commodities in $k \in U$. Each such flow equals the flow of that commodity k on arc $(r, j(s))$. For instance, the flow from node 1(3) to commodity 2 in the figure is shown to be $\frac{1}{3}$. This implies that $z_{r1(3)}^2 = \frac{1}{3}$.

Observe that in this example, the total flow into each commodity $k \in U$ is exactly 1. Furthermore, this partial solution satisfies (36a–39c). Hence, $(z, y) \in LP(z, y)$. From the figure, $s_{\min} = 2$, $s_{\max} = 3$, $\text{Com_Card}(2) = \{1, 4, 5, 6\}$ and $\text{Com_Card}(3) = \{1, 2, 3, 4, 5\}$. As well, $J_{\text{arc}}(2) = \{(r, 2(2)), (r, 3(2)), (r, 4(2)), (r, 5(2))\}$ and $J_{\text{arc}}(3) = \{(r, 1(3)), (r, 2(3))\}$. Applying *Algorithm* _{π_2} (*Algorithm* 4), in the first iteration with $\hat{s} = 2$, one obtains $\alpha(2) = \min \{ \lfloor |\text{Com_Card}(2)| / 2 \rfloor, |J_{\text{arc}}(\hat{s})| \} = 2$ and $\pi_2 = 2$. In steps 8 and 9, one obtains $\text{Count}_{\hat{s}^+}(1) = 2$, $\text{Count}_{\hat{s}^+}(4) = 2$, $\text{Count}_{\hat{s}^+}(5) = 2$ and $\text{Count}_{\hat{s}^+}(6) = 0$. Since $\hat{s} * \alpha(\hat{s}) = 4$, all commodities $\{1, 4, 5, 6\}$ are removed from $\text{Com_Card}(3)$, if they belong. Hence, $\text{Com_Card}(3) = \{2, 3\}$. In the next iteration, $\alpha(3) = \min \{ \lfloor \frac{2}{3} \rfloor, 2 \} = \min \{0, 2\} = 0$. Hence, $\pi_2 = \pi_2 + 0 = 2$. Since $\hat{s}$ has reached $s_{\max}$, the algorithm stops, with $\pi_2 = 2$. Here, the left-hand-side of (38) is 6, whereas the right-hand-side is $2 * (\frac{2}{3} + \frac{2}{3}) + (\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8}) + 2 = \frac{17}{3}$, that is this is a violation. It is easy to see that if *Algorithm* _{π_1} (*Algorithm* 3) is applied to the same example, then $\pi_1 = 3$ is obtained, resulting in the left-hand-side equal to $\frac{20}{3}$ and hence no violation.

4 | Computational Results

In this section, we present a detailed computational study comparing two formulations of the problem: the proposed extended formulation (P_{xy}) embedded within a tailored branch-and-cut framework, and the classical formulation (P_{xy}) solved using a generic branch-and-cut approach. Both models are implemented and solved using CPLEX 22.1.1.0, compiled with Microsoft Visual

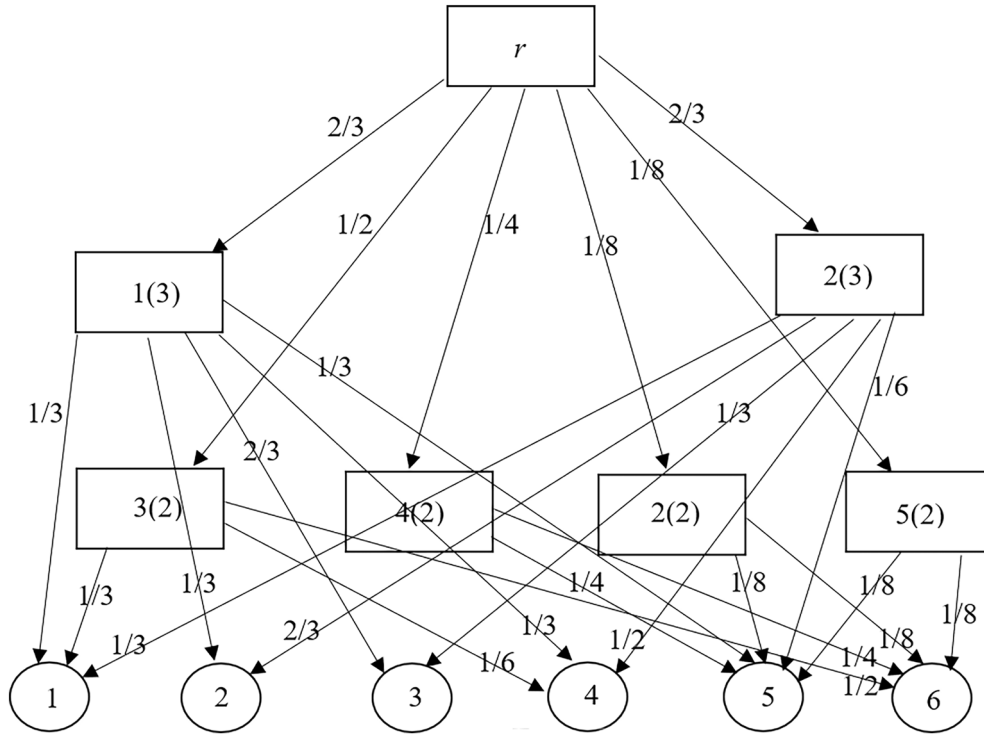


FIGURE 4 | Illustration of the fractional solution in Example (7).

C++ 2022, and tested on a machine equipped with a 2.9 GHz AMD Ryzen 74800H processor and 32 GB of RAM. The proposed branch-and-cut algorithm for (P_{zy}) incorporates the valid inequalities introduced in this paper. The separation algorithms for these inequalities, together with the extended model, are implemented in CPLEX using the Callable Library (C API) and are executed exclusively at the root node.

Two separate sets of datasets were used in this computational study. One set comprised a suitable modification of instances used by Beasley [34] for the closely related Steiner Tree Problem. These instances are defined on sparse graphs in which Steiner nodes are assumed to be uncapacitated. Thus, our modification involved adding direct connections from the root node to the customer nodes, adding capacities to the Steiner nodes, and demands to the terminal nodes. The second set of instances generated was on dense graphs. Furthermore, the costs associated with links that were generated are proportional to their capacity and length. The length of each link is determined as the Euclidean distance between the two nodes. For the sake of clarity in the subsequent exposition, we will henceforth refer to the former as CSAP-Sparse dataset, and the latter as CSAP-Dense dataset.

The Beasley [34] dataset consists of four sets (b, c, d, and e). A defining characteristic of these datasets are that the number of potential Steiner nodes are much larger than the number of terminal nodes. To adapt them to CSAP-Sparse, we randomly assigned capacities to the Steiner nodes within the range [40, 80] and modified the edge costs reported in the Beasley dataset by incorporating a component $F(C_j)$ related to the capacity of the corresponding Steiner node. More formally:

$$c_{ij} = \hat{c}_{ij}(1 - \rho) + \rho F(C_j) \quad (i, j) \in A, j \in S \quad (42)$$

$$c_{ij} = \hat{c}_{ij}(i, j) \in A, j \in T \quad (43)$$

in which $\hat{c}_{ij}$ is the cost associated with arc $(i, j) \in A$ reported in Beasley [34], ρ is the weighting factor and $F(C_j)$ is the fixed cost related to the capacity of Steiner node j , normalized with respect to the average cost between nodes. In our computational testing, ρ has been set to 0.8, while $F(C_j)$ is defined as:

$$F(C_j) := w \frac{\log(C_j)}{\log(\mathbb{E}_{s \in S}[C_s])} \mathbb{E}_{i, j \in N} [\hat{c}_{ij}] \quad (44)$$

where w is a weighting factor in $F(C_j)$ and is adopted to adjust the balance between the cost associated with capacities and the cost of selecting connections between nodes. $\mathbb{E}_{(s \in S)}[C_s]$ represents the average capacity of the Steiner nodes. The parameter w was set to 6. Finally, arcs that directly connect the root node to a terminal node are penalized with a cost that is thrice that of arcs connecting to Steiner nodes. This was done to make use of Steiner nodes more viable as part of the optimal solution. Additionally, demands for the terminal nodes were randomly generated within the range [25, 32].

Table A1 presents the comparison of bounds obtained at the root node between the classical formulation and the extended formulation on the CSAP-Sparse dataset, given a time limit of 1 h. In Table A1, the number of terminal and Steiner nodes is reported in columns “|T|” and “|S|”, respectively and the number of edges is reported in the column with heading “edges”. The column “gap” reports the percentage gap between the root node lower bound and the best-known integer solution and is computed as $\frac{BKS-LB}{LB}$ where BKS indicates the best-known solution and LB indicates the lower bound at the root node. By best-known solution we refer to the best integer solution found for the instance and not for

TABLE 1 | Comparison of average bounds obtained at the root node between the classical formulation and extended formulation on CSAP-sparse datasets (per-set and overall).

Name	Classical formulation					Extended formulation							
	Before cutting		After cutting			Before cutting		After cutting					
	#gaps	Gap (%)	Gap (%)	Time (s)	#CPX	#gaps	Gap (%)	Gap (%)	Time (s)	#cover	#1-pk	#match	#CPX
Set b	18/18	0.13	0.00	0.05	3.56	18/18	0.00	0.00	0.23	0.00	0.00	0.00	0.00
Set c	17/20	0.78	0.00	662.08	22.20	16/20	0.01	0.00	790.54	0.50	0.00	0.00	1.70
Set d	15/20	0.49	0.00	922.17	21.80	15/20	0.01	0.00	1130.85	0.00	0.75	0.00	0.40
Set e	12/20	0.21	0.00	1628.74	16.60	8/20	0.00	0.00	2258.14	0.00	0.00	0.00	7.55
Overall	62/78	0.41	0.00	823.85	16.36	57/78	0.00	0.00	1071.73	0.13	0.19	0.00	2.47

the formulation. Hence, it could be obtained by one of both the formulation. We also report the gaps in two points in time: before cutting and after cutting to evaluate the effectiveness of the cuts. The total CPU time (in seconds) taken to solve at the root node is also measured and reported in the columns with the heading “time”.

In the case of the classical formulation, CPLEX generates a variety of cuts using its separation heuristics. The number of such cuts added for each instance are reported under the heading “#CPX”. In the case of the extended formulation, we also report the number of violated (v, s, r) -cover, $(1, p_v)$ and the p -Arc Cardinality Matching inequalities, which are denoted under the heading “#cover”, “#1-pk” and “#match”, respectively. We use the label **TL** (Time Limit) to indicate instances where the solver was stopped after 1 h without proving optimality. The label **NF** indicates that a relative optimality gap could not be computed because at least one of its two components was missing within the imposed time limit. Specifically, either (i) a valid lower bound was unavailable because the LP relaxation at the root node did not solve within the 1-h limit, or (ii) no feasible incumbent solution was found before the time limit expired. For example, in the case of instance c_14, a feasible integer solution is known, as the classical formulation successfully produced one. However, the LP relaxation of the extended formulation required more than 1 h to solve at the root node, so no valid lower bound was obtained. As a result, the optimality gap for the extended formulation on instance c_14 could not be computed and is therefore reported as **NF**.

It is clear from the results shown in Table A1, that in most instances, the problems can be deemed as very easy, in that they were solved at the root node, both by the classical formulation and the extended formulation. It is indeed worth observing that in most instances, no cut was generated either for the classical formulation or the extended formulation. This is particularly the case with series b and to a large extent, series c, where the problem sizes are relatively smaller. One possible reason that we believe these problems to be easy is that the number of potential Steiner nodes are much more than the number of terminal nodes in most instances. For instance, e_1 consists of 2495 Steiner nodes, but only 5 terminal nodes. Thus, there is more than sufficient total capacity to meet the total demand of terminal nodes. Given the cost structure, relatively only a few Steiner nodes need to be utilized in the optimal solution. This then explains the relatively very small gap obtained from solving the LP relaxation of

the classical formulation. As the problem size increases, as with series d and e, finding a feasible integer solution, as well as a lower bound takes a long time. This problem becomes more acute with the extended formulation, since the number of variables and constraints are many times more than the case with the classical formulation. In all instances where the problem is solved at the root node by the classical formulation, the extended formulation takes more time to solve than the classical formulation. This is simply due to the extended formulation consisting of significantly more variables and constraints than the classical formulation. Therefore, then, the extended formulation offers no advantage over the classical formulation on CSAP-Sparse datasets. Table 1 shows the per-set and overall average of the results reported in Table A1. In addition, the columns labeled “#gaps” report the number of instances for which a gap could be computed that is an integer best known solution and a lower bound have been found. For instance, for 17 instances out of 20 of dataset c a gap could be computed.

CSAP-Dense data set is composed of sets a , b and c each with 20 instances. Set a consists of 35 Steiner nodes and 35 terminal nodes, instances in dataset b consists of 25 Steiner nodes and 45 terminal nodes, and instances in dataset c consists of 20 Steiner nodes and 50 terminal nodes. It is assumed, without loss of generality, that there exists a directed arc (r, j) from root node r to every node $j \in \{r\} \cup S$. For arcs that do not exist, their costs are assigned an arbitrarily large value. Similarly, the sub-graph, $G' = (S, A')$, induced by S is fully connected, where $A' = \{(i, j) \in A \mid i \in \{r\} \cup S, j \in S\}$. Finally, there exists an arc from each node $i \in S$ to each node $j \in T$. Thus, each instance consists of $(|S| + 1) \times |T| + (|S| \times |S|)$ arcs and therefore classified as dense graphs.

The cost associated with arcs (i, j) , where $i \in \{r\} \cup S$ and $j \in S$, is obtained as a linear combination of the Euclidean distance between nodes i and j , and a cost proportional to the capacity of the Steiner node. This clearly reflects the case with supply chain and telecommunication networks, where the cost increases with both capacity and length of the link. On the other hand, the costs of arcs connecting nodes in $\{r\} \cup S$ to nodes in T are purely a function of its distance as each such arc has sufficient capacity to meet the demand of that node in T . More formally:

$$c_{ij} = \text{dist}_{ij}(1 - \rho) + \rho F(C_j) \quad i \in \{r\} \cup S, j \in S \quad (45)$$

$$c_{ij} = \text{dist}_{ij} \quad i \in \{r\} \cup S, j \in T \quad (46)$$

TABLE 2 | Comparison of average bounds obtained at the root node between the classical formulation and extended formulation on CSAP-Dense datasets (per-set and overall).

Name	Classical formulation					Extended formulation							
	Before cutting		After cutting			Before cutting		After cutting					
	#gaps	Gap (%)	Gap (%)	Time (s)	#CPX	#gaps	Gap (%)	Gap (%)	Time (s)	Cover	1-pk	Match	#CPX
Set a	20/20	5.82	0.41	72.56	151.25	20/20	1.74	0.13	145.25	18.80	7.10	0.95	50.60
Set b	20/20	6.60	0.28	49.89	172.20	20/20	1.38	0.11	86.27	18.65	12.00	3.75	59.40
Set c	20/20	61.83	10.94	37.40	397.95	20/20	12.73	4.59	26.52	15.40	8.15	1.55	71.40
Overall	60/60	24.75	3.87	53.28	240.46	60/60	5.28	1.61	86.01	17.62	9.08	2.08	60.47

in which $dist_{ij}$ represents the Euclidean distance between nodes, ρ is the weighting factor and $F(C_j)$ is the fixed cost related to the capacity of Steiner node j , normalized with respect to the average distance between nodes. In our computational testing, ρ has been set to 0.8, while $F(C_j)$ is defined as:

$$F(C_j) := w \frac{\log(C_j)}{\log(\mathbb{E}_{s \in S}[C_s])} \mathbb{E}_{i,j \in N}[dist_{ij}] \quad (47)$$

where w is a weighting factor in $F(C_j)$ and is adopted to adjust the balance between the cost associated with capacities and the cost of selecting connections between nodes. $\mathbb{E}_{s \in S}[C_s]$ represents the average capacity of the Steiner nodes. In our experiments, w was set to 6. The idea behind these cost settings is that arcs ending at Steiner nodes having higher capacity are charged more on average, just as arcs covering larger distance cost more. Finally, arcs that directly connect the root node to a customer node are penalized with a cost that is approximately thrice that of arcs connecting to Steiner nodes. This was done to make use of Steiner nodes more viable as part of the optimal solution. For all instances, the capacity C_j of each Steiner node j was randomly generated in the range $[40, 80]$ using a uniform distribution. Similarly, the demand associated with each $j \in T$ was generated in the range $[20, 30]$ using a uniform distribution. Computational testing was conducted on the 60 randomly generated test problems. The results are presented in Table A2, which is reported along the same lines as in Table A1.

Several features are apparent from the results in Table A2. The first is that the bounds obtained from the initial LP relaxation of the classical formulation are poor. The percentage gap varies from 4.4% (in instance a_4 and a_13) to 217.9% (in instance c_1), with an average of 24.75%. In contrast, bounds obtained from the initial LP relaxation of the extended formulation are very tight, ranging from 0.20% (in instance c_11 and c_19) to 37.3% (in instance c_1) with an average of 5.28%. Consequently, for the classical formulation, many cuts were generated by CPLEX. These numbers varied from 62 in instance a_17 to 882 cuts in instance c_2 . Computationally, this translates to that many more re-optimizations of the resulting LPs. In contrast, significantly fewer cuts were generated for the extended formulation. Due to the large number of CPLEX cuts in the classical formulation, the time taken at the root node is comparable to that of the extended formulation, despite the extended formulation being much larger than the classical formulation. The most important observation is that with a fewer number of total cuts, the extended formulation can close the gap after the addition of cuts much more effectively

than the classical formulation. With the extended formulation, the optimal integer solution was obtained at the root node in 25 of the 60 problem instances. The percentage gap was found to be less than 2% in 27 of the remaining 35 test problems, in which an optimal integer solution was not found at the root node of the branch-and-bound tree. Note that the optimal integer solution was found at the root node in just 8 instances using the classical formulation. Of the 52 remaining problem instances, 37 had a percentage gap below 2%.

The results presented in Table A2 also show that the percentage gap at the root node of the branch-and-bound tree in data set c is significantly more than that in sets a and b . Thus, data set c represents instances that are much more difficult to solve to optimality than instances in sets a and b . Problem instances in data set a are seen to be easier than those in b . It is readily apparent from this observation that with the range of capacities of Steiner nodes and demands of terminal nodes being the same, as the ratio of the number of terminal nodes to Steiner nodes increases, the problem instances tend to become harder. This is because it becomes harder to accommodate the demands of terminal nodes with the available Steiner nodes. In fact, in some instances, with insufficient capacities on Steiner nodes, direct connections from the root node to some of the terminal nodes may be necessary.

All the test problems listed in Table A2 were made to run to optimality on both the formulations, that is the CPLEX generated branch-and-cut on the classical formulation, and the CPLEX generated branch-and-cut on the extended formulation, along with user generated cuts of $(v, s, \hat{r}_s)$ -cover, $(1, \hat{p}_v)$ -configuration and the p -Arc Cardinality Matching inequalities. For both formulations, all generated cuts are limited to the root node of the branch-and-bound tree. Table 2 shows the per-set and overall average of the results in Table A2. In addition, the columns labeled “#gaps” report the number of instances for which a gap could be computed that is an integer best known solution and a lower bound have been found. For instance, for 20 instances out of 20 of dataset c a gap could be computed.

The number of nodes required by the branch-and-cut procedure to solve the problem instances to optimality, as well as the total time taken are presented in Table A3.

It is quite evident from the results in Table A3 that in those relatively ‘easy’ instances, where the number of branch-and-bound nodes required using the classical formulation is not large, the time taken by it to solve the instance to optimality is marginally

TABLE 3 | Comparison of average number of nodes in the branch-and-cut tree and the average total computational time between the classical formulation and extended formulation on CSAP-Dense datasets (per-set and overall).

Name	Classical formulation			Extended formulation		
	#solved	Nodes	Time (s)	#solved	Nodes	Time (s)
Set a	20/20	2338.45	249.91	20/20	20.75	158.83
Set b	19/20	3524.80	298.58	20/20	31.95	100.78
Set c	11/20	67591.50	1767.68	20/20	98.15	48.84
Overall	50/60	24484.91	772.05	60/60	50.28	102.82

TABLE 4 | Average performance metrics for the extended formulation on CSAP-Dense instances (per-set and overall), along with root-node bounds before and after cutting, number of CPLEX cuts added, branch-and-bound nodes, and total computational time.

Name	Before cutting		After cutting					
	#gaps	Gap (%)	#gaps	Gap (%)	Time (s)	#CPX	Nodes	Time (s)
Set a	20/20	1.74	20/20	0.17	160.46	56.75	199.15	181.72
Set b	20/20	1.38	20/20	0.16	107.92	72.10	73.00	121.85
Set c	20/20	12.73	20/20	4.58	31.92	106.00	576.15	65.25
Overall	60/60	5.28	60/60	1.64	100.10	78.28	282.77	122.94

better than the time taken by the extended formulation. This is because the extended formulation is considerably larger in size and therefore requires more time to solve each relaxation even though it consistently requires fewer nodes than the classical formulation. Thus, in some instances the extended formulation achieves better times, whereas in others, specifically, a_3 , a_4 , a_6 , a_7 , a_{11} , a_{12} , a_{13} , a_{15} , a_{16} , and a_{17} , the classical formulation solves the problem faster by exploiting relatively few branch-and-bound nodes. Overall, however, the solution times of the two formulations remain comparable. This is particularly so in the case of those instances of set a . Here, the times taken by the classical formulation varied from 70.39 s, instance a_{13} , to 620.26 s. In comparison, the times taken by the extended formulation varied from 92.0 s to 254.2 s. In terms of number of branch-and-bound nodes, the classical formulation required nodes varying from 0 to 20 784, whereas the extended formulation required between 0 and 223. It must be noted that even in those instances of dataset a where the extended formulation took more time, the difference was only marginal.

However, the results are dramatically different in those ‘hard’ problem instances where the gap between the lower bound and upper bound obtained using the classical formulation is relatively larger. This seems to be the case with sets b and c . In set b , the performance of the extended formulation in comparison to the classical formulation is even better. The times taken by the classical formulation varied from 44.87 s to over 3600 s. Note also that problem b_{18} could not be solved to optimality even after 3600 s. In contrast, the time taken by the extended formulation varied from 20.3 s to 165.2 s. Here, the extended formulation took less time than the classical formulation in 8 of the 20 instances. In terms of the number of nodes required, the extended formulation uniformly outperformed the classical formulation. It is worth noting that, while the classical formulation could not solve the b_{18} even after 60 585 nodes, the same was solved by the extended formulation in 29 nodes.

The results obtained from dataset c series are even more dramatic. For problem instances c_1 , c_3 , c_5 , c_9 , c_{12} , c_{13} , c_{16} , c_{17} and c_{18} the classical formulation could not solve in 3600 s, with almost all of them requiring more than 100 000 nodes. These same problems were solved using relatively few branch-and-bound nodes by the extended formulation. Of these, c_1 required the most branch-and-bound nodes (i.e., 315) by the extended formulation. It is also interesting to note that c_3 was solved at the root node by the extended formulation. Consequently, the times taken by the extended formulation to solve these ‘hard’ instances was found to be many orders of magnitude faster than those taken by the classical formulation. The most time taken by the extended formulation to solve any instance in the c dataset series is 96.6 s. In comparison, many problems in the c set series were not solved to optimality even after 3600 s by the classical formulation. This then is the inherent value of using the extended formulation along with its associated valid inequalities. Table 3 shows the per-set and overall average of the results reported in Table A3. In addition, the columns labeled “#solved” report the number of instances solved by each formulation. For instance, 11 instances out of 20 of dataset c are solved by the classical formulation, whereas all instances are solved by the extended formulation. It is amply clear from Table 3 that the extended formulation outperforms the classical formulation by a huge margin, both in terms of time taken and the number of branch-and-bound nodes needed to establish optimality.

To better understand the unique contribution of our user generated cuts for the extended formulation on the overall performance, we report in Table A4 the bounds obtained at the root node using the extended formulation on the generated instances, without our cut procedures. Table 4 shows the per-set and overall average of the results reported in Table A4. Comparing the results in Table A4 to that in Table A2, it is evident that our cut procedures lead to a better bound in most instances (e.g., c_2), whereas in a few others the bound worsens (e.g., c_1). This may occur

because, in certain cases, the default CPLEX cuts interact more effectively with the problem structure when used alone. On average, however, the gap obtained after applying the cuts is smaller when our cut procedures are employed. More significantly, the average number of nodes and the average total computational time taken to solve the instances is substantially lower when using our cut procedures. Therefore, there is significant value in incorporating our cut procedures in addition to those generated by CPLEX on the extended formulation.

5 | Concluding Remarks

In this paper, we present a new extended formulation of CSAP that uses the idea of cardinality associated with each Steiner node. Even though the resulting formulation is bigger in terms of the number of variables and constraints, we show theoretically that it is indeed a stronger formulation, that is its LP relaxation provides a tighter lower bound than the LP relaxation of the classical formulation. In addition, we show that our proposed extended formulation reveals generalizations of the Cover and $(1, k)$ -Configuration inequalities which collectively are stronger than the original Cover and $(1, k)$ -Configuration inequalities. Finally, we present another class of inequalities called p -Arc Cardinality Matching Inequalities, which are specific to the extended formulation.

We present results of extensive computational tests on data instances reported in Beasley [34], as well as on 60 randomly generated problems that we generated on dense graphs with specific characteristics. The datasets reported in Beasley [34] were suitably modified to incorporate capacity and demand data. These graphs are sparse, and hence we refer to them in our test as CSAP-Sparse. In most instances, these problems were found to be very easy, with the problem instances being solved at the root node itself, both by the classical formulation and the extended formulation. For a few instances, where the problem size was big, the extended formulation could not establish bounds as it took a long time to solve the LP at the root node. This is because the extended formulation is substantially larger in terms of number of variables and constraints.

We next generated 60 problem instances on dense graphs, which we refer to as CSAP-Dense. Further, the cost data generated was such that they increase with increasing capacity, as well as increase with the length of the arc. This feature subscribes to real-world supply chain and telecommunication networks. What is indeed apparent from this dataset is that using the classical formulation, the gap between upper and lower bound at the root node of the branch-and-bound tree is substantially more than that obtained using the extended formulation. Consequently, the number of nodes required by using the extended formulation to establish optimality was orders of magnitude less than that obtained from using the classical formulation. The results on CSAP-Dense clearly confirm the value of using our proposed extended formulation, that is it provides a robust alternative to the classical formulation. For those problems that the classical formulation solves reasonably quickly, the extended formulation also solves them with lesser number of nodes, even though it takes slightly more time due to its problem size being larger. However, for those ‘hard’ problem instances that the classical

formulation is unable to solve in reasonable time, the extended formulation solves them many orders of magnitude faster. We were able to demonstrate that our approach was able to identify the optimal solution at the root node itself in many instances. Furthermore, for all those problem instances that the classical formulation was unable to solve in 1 h, those same problems were solved in less than 2 min using the extended formulation. This clearly demonstrates the value of using the extended formulation proposed in this paper. The results in this paper, both theoretical and computational, demonstrate an approach that seems very promising for the more general version of CSAP, which is the capacitated network design problem involving non-splittable multiple commodities, in which each commodity requires a route from its source to its destination.

Author Contributions

Francesco Contu: validation, software, data curation, writing – review and editing. **Massimo Di Francesco:** investigation, supervision, validation, writing – review and editing. **Enrico Gorgone:** investigation, supervision, validation, writing – review and editing. **Ishwar Murthy:** conceptualization, supervision, writing – review and editing, writing – original draft.

Acknowledgments

This study was carried out within the MOST—Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR)—MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4—D.D. 1033 17/06/2022, CN00000023). This manuscript reflects only the authors’ views and opinions, neither the European Union nor the European Commission can be considered responsible for them. This publication was produced by Francesco Contu while attending the PhD programme in Mathematics and Computer Science at the University of Cagliari, Cycle XXXVIII, with the support of a scholarship co-financed by the Ministerial Decree no 352 of 9th April 2022 based on the NRPP—funded by the European Union—Next Generation EU-Mission 4 “Education and Research”, Component 2 “From Research to Business”, Investment 3.3 and by the company Spiva. Open access publishing facilitated by Università degli Studi di Cagliari, as part of the Wiley - CRUI-CARE agreement.

Funding

This work was supported by NRPP—funded by the European Union—Next Generation EU-Mission 4.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. M. Filipecki and M. Van Vyve, “Stronger Path-Based Extended Formulation for the Steiner Tree Problem,” *Networks* 75, no. 1 (2020): 3–17.
2. S. Van den Eynde, P. Audenaert, D. Colle, and M. Pickavet, “A Construction Heuristic for the Capacitated Steiner Tree Problem,” *PLoS One* 17 (2022): 1–17.
3. R. T. Wong, “A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph,” *Mathematical Programming* 28 (1984): 271–287.
4. M. X. Goemans and Y.-S. Myung, “A Catalog of Steiner Tree Formulations,” *Networks* 23 (1993): 19–28.

5. T. Polzin and S. V. Daneshmand, "A Comparison of Steiner Tree Relaxations," *Discrete Applied Mathematics* 112 (2001): 241–261.
6. C. W. Duin, *Steiner's Problem in Graphs* (Amsterdam University, 1993).
7. M. Charikar, C. Chekuri, T. Y. Cheung, et al., "Approximation Algorithms for Directed Steiner Problems," *Journal of Algorithms* 33, no. 1 (1999): 73–91.
8. B. Gavish, "Formulations and Algorithms for the Capacitated Minimal Directed Tree Problem," *Journal of the ACM (JACM)* 30, no. 1 (1983): 118–132.
9. L. Gouveia and P. Martins, "The Capacitated Minimal Spanning Tree Problem: An Experiment With a Hop-Indexed model," *Annals of Operations Research* 86 (1999): 271–294.
10. L. Gouveia and M. J. Lopes, "The Capacitated Minimum Spanning Tree Problem: On Improved Multistar Constraints," *European Journal of Operational Research* 160, no. 1 (2005): 47–62.
11. L. Gouveia, "A 2n Constraint Formulation for the Capacitated Minimal Spanning Tree Problem," *Operations Research* 43, no. 1 (1995): 130–141.
12. X. Cheng and D. Z. Du, *Steiner Trees in Industry*, vol. 11 (Springer Science & Business Media, 2013).
13. F. K. Hwang, D. S. Richards, and P. Winter, "The Steiner Tree Problem," *Annals of Discrete Mathematics* 53 (1992): 55–89.
14. I. Ljubić, "Solving Steiner Trees: Recent Advances, Challenges, and Perspectives," *Networks* 77, no. 2 (2021): 177–204.
15. S. L. Hakimi, "Steiner's Problem in Graphs and Its Implications," *Networks* 1, no. 2 (1971): 113–133.
16. W. Liu, "A Lower Bound for the Steiner Tree Problem in Directed Graphs," *Networks* 20 (1990): 765–778.
17. S. Chopra, E. R. Gorres, and M. R. Rao, "Solving the Steiner Tree Problem on a Graph Using Branch and Cut," *ORSA Journal of Computing* 4, no. 3 (1992): 320–335.
18. T. Polzin and S. V. Daneshmand, "Approaches to the Steiner Problem in Networks," in *Algorithmics of Large and Complex Networks*, ed. J. Lerner, D. Wagner, and K. A. Zweig (Springer, 2009), 81–103.
19. D. M. Warme, *Spanning Trees in Hypergraphs With Applications to Steiner Trees* (University of Virginia, 1998).
20. D. Rehfeldt and T. Koch, "Implications, Conflicts, and Reductions for Steiner Trees," *Mathematical Programming* 197, no. 2 (2023): 903–966.
21. C. W. Duin and A. Volgenant, "An Edge Elimination Test for the Steiner Problem in Graphs," *Operations Research Letters* 8, no. 2 (1989): 79–83.
22. C. Bentz, M.-C. Costa, and A. Hertz, "On the Edge Capacity Steiner Tree Problem," *Discrete Optimization* 38 (2020): 1–25.
23. L. Di Puglia Pugliese, M. Gaudioso, F. Guerriero, and G. Miglionico, "A Lagrangean-Based Decomposition Approach for the Link Constrained Steiner Tree Problem," *Optimization Methods and Software* 33 (2018): 650–670.
24. C. Bentz, M.-C. Costa, P. L. Poirion, and T. Ridremont, "Robust Capacitated Steiner Trees and Networks With Uniform Demands," *Networks* 82, no. 1 (2023): 3–31.
25. A. Segev, "The Node-Weighted Steiner Tree Problem," *Networks* 17, no. 1 (1987): 1–17.
26. D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson, "A Note on the Prize Collecting Traveling Salesman Problem," *Mathematical Programming* 59, no. 1 (1993): 413–420.
27. M. Leitner, I. Ljubić, M. Luipersbeck, and M. Sinnl, "A Dual Ascent-Based Branch-And-Bound Framework for the Prize-Collecting Steiner Tree and Related Problems," *INFORMS Journal on Computing* 30, no. 2 (2018): 402–420.
28. D. Rehfeldt and T. Koch, "On the Exact Solution of Prize-Collecting Steiner Tree Problems," *INFORMS Journal on Computing* 34, no. 2 (2022): 872–889.
29. I. Murthy and S. Ransbotham, "A New Extended Formulation of the Generalized Assignment Problem and Some Associated Valid Inequalities," *Discrete Applied Mathematics* 271 (2019): 119–143.
30. M. Di Francesco, M. Gaudioso, E. Gorgone, and I. Murthy, "A New Extended Formulation With Valid Inequalities for the Capacitated Concentrator Location Problem," *European Journal of Operational Research* 289, no. 3 (2021): 975–986.
31. IBM Corporation, "IBM ILOG CPLEX Optimizer (Version 22.1.1.0) [Computer Software]. IBM," (2023), <https://www.ibm.com/docs/en/icos/22.1.1>.
32. T. Koch and A. Martin, "Solving Steiner Tree Problems in Graphs to Optimality," *Networks* 32 (1998): 207–232.
33. M.-W. Padberg, "(1, k)-Configurations and Facets of Packing Problems," *Mathematical Programming* 18 (1980): 94–99.
34. J. E. Beasley, "An SST-Based Algorithm for the Steiner Problem in Graphs," *Networks* 19 (1989): 1–16.

Appendix A

TABLE A1 | Comparison of bounds obtained at the root node between the classical formulation and extended formulation on CSAP-sparse datasets.

Problems				Classical formulation				Extended formulation						
				Before cutting	After cutting			Before cutting	After cutting					
					Gap (%)	Gap (%)	Time (s)		CPX	Gap (%)	Gap (%)	Time (s)	#cover	#1-pk
b_1	41	9	63	0.00	0.00	0.01	0	0.00	0.00	0.05	0	0	0	0
b_2	37	13	63	0.00	0.00	0.01	0	0.00	0.00	0.03	0	0	0	0
b_3	25	25	63	0.00	0.00	0.01	0	0.00	0.00	0.06	0	0	0	0
b_4	41	9	100	0.00	0.00	0.01	0	0.00	0.00	0.11	0	0	0	0
b_5	37	13	100	0.00	0.00	0.01	0	0.00	0.00	0.08	0	0	0	0
b_6	25	25	100	0.00	0.00	0.02	16	0.00	0.00	0.06	0	0	0	0
b_7	62	13	94	0.00	0.00	0.01	0	0.00	0.00	0.06	0	0	0	0
b_8	56	19	94	0.00	0.00	0.01	0	0.00	0.00	0.06	0	0	0	0
b_9	37	38	94	0.00	0.00	0.08	16	0.00	0.00	0.08	0	0	0	0
b_10	62	13	150	0.00	0.00	0.01	0	0.00	0.00	0.19	0	0	0	0
b_11	56	19	150	0.00	0.00	0.01	0	0.00	0.00	0.23	0	0	0	0
b_12	37	38	150	0.00	0.00	0.01	0	0.00	0.00	0.22	0	0	0	0
b_13	83	17	125	0.00	0.00	0.01	0	0.00	0.00	0.28	0	0	0	0
b_14	75	25	125	0.00	0.00	0.02	0	0.00	0.00	0.42	0	0	0	0
b_15	50	50	125	2.30	0.00	0.29	20	0.00	0.00	0.19	0	0	0	0
b_16	83	17	200	0.00	0.00	0.04	12	0.00	0.00	0.71	0	0	0	0
b_17	75	25	200	0.00	0.00	0.05	0	0.00	0.00	1.05	0	0	0	0
b_18	50	50	200	0.00	0.00	0.29	0	0.00	0.00	0.27	0	0	0	0
c_1	495	5	625	0.00	0.00	0.02	0	0.00	0.00	0.93	0	0	0	0
c_2	490	10	625	0.00	0.00	0.04	0	0.00	0.00	3.15	0	0	0	0
c_3	417	83	625	0.00	0.00	1.02	9	0.00	0.00	25.27	0	0	0	0
c_4	375	125	625	0.00	0.00	0.92	0	0.00	0.00	42.70	0	0	0	0
c_5	250	250	625	0.70	0.00	672.56	51	0.00	0.00	22.04	0	0	0	0
c_6	495	5	1000	0.00	0.00	0.03	0	0.00	0.00	2.51	0	0	0	0
c_7	490	10	1000	0.00	0.00	0.06	0	0.00	0.00	5.06	0	0	0	0
c_8	417	83	1000	0.00	0.00	7.22	36	0.00	0.00	150.60	0	0	0	0
c_9	375	125	1000	0.00	0.00	1.60	0	0.00	0.00	103.31	0	0	0	0
c_10	250	250	1000	2.70	0.00	761.10	100	0.00	0.00	41.77	0	0	0	0
c_11	495	5	2500	0.00	0.00	0.06	0	0.00	0.00	0.89	0	0	0	0
c_12	490	10	2500	0.00	0.00	0.17	0	0.00	0.00	4.21	0	0	0	0
c_13	417	83	2500	0.10	0.00	250.33	50	0.00	0.00	135.96	1	0	0	9
c_14	375	125	2500	6.90	0.00	442.99	63	NF	NF	TL	0	0	0	0
c_15	250	250	2500	2.10	0.00	17.50	129	0.10	0.00	736.61	9	0	0	25
c_16	495	5	12 500	0.00	0.00	52.19	0	0.00	0.00	18.71	0	0	0	0
c_17	490	10	12 500	0.80	0.00	233.72	6	0.00	0.00	117.07	0	0	0	0
c_18	417	83	12 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0
c_19	375	125	12 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0
c_20	250	250	12 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0

(Continues)

TABLE A1 | (Continued)

Problems				Classical formulation				Extended formulation						
				Before cutting	After cutting			Before cutting	After cutting					
					Gap (%)	Gap (%)	Time (s)		CPX	Gap (%)	Gap (%)	Time (s)	#cover	#1-pk
d_1	995	5	1250	0.00	0.00	0.04	0	0.00	0.00	3.41	0	0	0	0
d_2	990	10	1250	0.00	0.00	0.09	0	0.00	0.00	8.66	0	0	0	0
d_3	833	167	1250	0.00	0.00	5.42	0	0.00	0.00	334.55	0	0	0	0
d_4	750	250	1250	0.60	0.00	6.59	38	0.00	0.00	220.28	0	0	0	0
d_5	500	500	1250	0.90	0.00	14.46	35	0.00	0.00	121.98	0	0	0	0
d_6	995	5	2000	0.00	0.00	0.05	0	0.00	0.00	4.63	0	0	0	0
d_7	990	10	2000	0.00	0.00	0.13	0	0.00	0.00	13.20	0	0	0	0
d_8	833	167	2000	0.00	0.00	8.25	8	0.00	0.00	844.02	0	0	0	0
d_9	750	250	2000	0.80	0.00	13.94	68	0.00	0.00	1439.31	0	0	0	0
d_10	500	500	2000	0.60	0.00	23.16	72	0.10	0.00	395.89	0	0	0	0
d_11	995	5	5000	0.00	0.00	0.10	0	0.00	0.00	2.76	0	0	0	0
d_12	990	10	5000	0.00	0.00	0.21	0	0.00	0.00	4.22	0	0	0	0
d_13	833	167	5000	NF	NF	TL	0	NF	NF	TL	0	0	0	0
d_14	750	250	5000	NF	NF	TL	0	0.00	0.00	977.09	0	15	0	8
d_15	500	500	5000	4.50	0.00	209.81	215	NF	NF	TL	0	0	0	0
d_16	995	5	25 000	0.00	0.00	25.06	0	0.00	0.00	55.92	0	0	0	0
d_17	990	10	25 000	0.00	0.00	136.07	0	0.00	0.00	191.07	0	0	0	0
d_18	833	167	25 000	NF	NF	TL	0	NF	NF	TL	0	0	0	0
d_19	750	250	25 000	NF	NF	TL	0	NF	NF	TL	0	0	0	0
d_20	500	500	25 000	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_1	2495	5	3125	0.00	0.00	1.48	0	0.00	0.00	20.70	0	0	0	0
e_2	2490	10	3125	0.00	0.00	4.60	0	0.00	0.00	51.93	0	0	0	0
e_3	2083	417	3125	0.00	0.00	904.47	20	NF	NF	TL	0	0	0	14
e_4	1875	625	3125	0.30	0.00	1115.87	50	NF	NF	TL	0	0	0	37
e_5	1250	1250	3125	0.80	0.00	1006.44	138	NF	NF	TL	0	0	0	100
e_6	2495	5	5000	0.00	0.00	0.18	0	0.00	0.00	61.20	0	0	0	0
e_7	2490	10	5000	0.00	0.00	0.40	0	0.00	0.00	206.93	0	0	0	0
e_8	2083	417	5000	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_9	1875	625	5000	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_10	1250	1250	5000	0.90	0.00	518.73	115	NF	NF	TL	0	0	0	0
e_11	2495	5	12 500	0.00	0.00	0.24	0	0.00	0.00	5.49	0	0	0	0
e_12	2490	10	12 500	0.00	0.00	0.60	0	0.00	0.00	20.93	0	0	0	0
e_13	2083	417	12 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_14	1875	625	12 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_15	1250	1250	12 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_16	2495	5	62 500	0.00	0.00	2.28	0	0.00	0.00	424.36	0	0	0	0
e_17	2490	10	62 500	2.20	0.00	219.57	9	0.00	0.00	1171.18	0	0	0	0
e_18	2083	417	62 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_19	1875	625	62 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0
e_20	1250	1250	62 500	NF	NF	TL	0	NF	NF	TL	0	0	0	0

TABLE A2 | Comparison of bounds obtained at the root node between the classical formulation and extended formulation on CSAP-Dense datasets.

Classical formulation								Extended formulation						
Problems				Before cutting	After cutting			Before cutting	After cutting					
Name	S	T	Edges	Gap (%)	Gap (%)	Time (s)	#CPX	Gap (%)	Gap (%)	Time (s)	#cover	#1-pk	#match	#CPX
a_0	35	35	2485	6.30	0.50	84.23	230	2.10	0.40	158.40	15	16	0	60
a_1	35	35	2485	6.50	0.70	85.67	190	1.80	0.00	142.10	18	0	0	38
a_2	35	35	2485	7.30	0.30	95.42	119	2.20	0.00	158.60	27	0	0	56
a_3	35	35	2485	5.30	0.50	83.89	91	1.90	0.40	202.70	36	0	1	54
a_4	35	35	2485	4.40	0.10	67.63	96	1.60	0.00	139.50	31	10	0	68
a_5	35	35	2485	6.50	0.20	80.55	109	1.10	0.10	103.50	10	17	1	50
a_6	35	35	2485	5.50	0.20	82.35	100	1.70	0.10	179.80	31	5	5	55
a_7	35	35	2485	5.70	0.20	65.42	87	0.80	0.00	91.10	11	0	4	33
a_8	35	35	2485	5.50	0.80	57.33	216	1.50	0.00	132.00	8	15	0	45
a_9	35	35	2485	6.10	0.00	77.73	87	1.80	0.00	152.30	17	0	0	46
a_10	35	35	2485	5.80	1.00	101.76	487	2.60	0.20	234.90	18	20	4	68
a_11	35	35	2485	7.60	0.20	76.06	97	1.30	0.10	156.20	23	3	0	38
a_12	35	35	2485	6.10	0.20	76.51	105	2.30	0.10	161.10	19	4	0	54
a_13	35	35	2485	4.40	0.10	57.95	85	1.00	0.10	115.40	11	5	0	35
a_14	35	35	2485	4.50	0.70	50.49	186	1.70	0.00	196.50	19	0	0	34
a_15	35	35	2485	7.30	0.40	55.35	108	2.40	0.20	115.89	23	6	2	56
a_16	35	35	2485	4.90	0.50	61.67	110	2.20	0.40	134.44	21	8	2	70
a_17	35	35	2485	4.90	0.20	60.46	62	0.60	0.10	99.60	10	17	0	48
a_18	35	35	2485	6.90	0.00	80.59	104	1.60	0.00	71.60	15	0	0	50
a_19	35	35	2485	4.80	1.30	50.05	356	2.50	0.40	159.38	13	16	0	54
b_0	25	45	3175	10.20	0.50	60.05	262	2.20	0.20	113.10	25	11	5	71
b_1	25	45	3175	7.20	0.40	56.88	130	1.90	0.20	114.30	12	19	11	49
b_2	25	45	3175	6.20	0.10	67.44	144	1.30	0.00	85.60	23	1	0	68
b_3	25	45	3175	6.80	0.50	45.14	241	1.10	0.00	84.30	9	10	11	39
b_4	25	45	3175	5.70	0.00	46.40	138	2.00	0.00	130.10	28	0	0	62
b_5	25	45	3175	5.20	0.30	42.77	180	1.40	0.20	83.90	32	33	17	57
b_6	25	45	3175	6.40	0.30	47.01	144	0.90	0.00	94.70	13	36	3	66
b_7	25	45	3175	6.40	0.40	49.10	181	1.40	0.40	67.40	20	20	0	95
b_8	25	45	3175	6.50	0.10	40.74	94	0.60	0.10	59.70	7	18	0	46
b_9	25	45	3175	8.40	0.10	54.25	125	1.20	0.10	56.40	19	0	0	54
b_10	25	45	3175	5.30	0.40	51.29	262	1.30	0.30	87.10	18	11	4	79
b_11	25	45	3175	5.90	0.30	50.73	161	1.30	0.10	141.30	20	16	1	68
b_12	25	45	3175	6.30	0.00	56.03	106	1.60	0.00	84.10	32	5	0	72
b_13	25	45	3175	5.50	0.20	49.09	131	1.60	0.10	75.70	30	15	0	56
b_14	25	45	3175	8.50	0.50	54.85	218	1.20	0.00	79.40	9	6	0	41
b_15	25	45	3175	5.10	0.10	31.25	112	1.20	0.10	97.70	19	1	5	71
b_16	25	45	3175	6.10	0.00	37.66	77	0.30	0.00	21.50	4	1	0	37
b_17	25	45	3175	7.50	0.20	66.94	119	2.10	0.10	80.80	20	12	5	50
b_18	25	45	3175	7.10	0.90	34.72	461	1.50	0.30	56.20	8	18	11	41
b_19	25	45	3175	5.70	0.30	55.44	158	1.50	0.00	112.10	25	7	2	66
c_0	20	50	3520	29.80	0.10	23.38	134	17.30	3.50	22.40	25	5	4	139
c_1	20	50	3520	217.90	47.50	34.03	330	37.30	36.40	17.10	16	17	3	80
c_2	20	50	3520	133.30	18.20	49.51	882	20.30	0.20	28.90	31	5	0	81
c_3	20	50	3520	32.70	8.00	34.93	675	8.40	0.00	24.70	5	6	0	49
c_4	20	50	3520	28.80	0.00	27.80	109	19.10	0.00	16.50	11	7	0	66
c_5	20	50	3520	21.70	2.70	39.18	672	2.50	0.00	33.30	14	0	0	67
c_6	20	50	3520	132.00	32.80	32.39	109	33.50	0.00	16.50	9	0	0	48

(Continues)

TABLE A2 | (Continued)

Problems				Classical formulation				Extended formulation						
				Before cutting	After cutting			Before cutting	After cutting					
					Gap (%)	Gap (%)	Time (s)		#CPX	Gap (%)	Gap (%)	Time (s)	#cover	#1-pk
Name	S	T	Edges	Gap (%)	Gap (%)	Time (s)	#CPX	Gap (%)	Gap (%)	Time (s)	#cover	#1-pk	#match	#CPX
c_7	20	50	3520	35.50	6.90	34.55	497	6.10	5.80	13.80	8	9	6	56
c_8	20	50	3520	68.60	0.00	37.37	111	10.90	5.80	18.30	7	18	8	49
c_9	20	50	3520	93.80	21.80	41.87	656	21.50	14.40	22.80	13	9	0	69
c_10	20	50	3520	4.50	0.00	43.99	106	0.60	0.00	41.30	17	0	0	61
c_11	20	50	3520	23.20	0.00	29.43	88	0.20	0.00	14.50	13	1	0	60
c_12	20	50	3520	72.90	10.50	45.76	676	7.70	6.30	22.10	12	0	6	67
c_13	20	50	3520	72.50	12.30	48.62	494	11.70	7.30	27.90	18	3	0	78
c_14	20	50	3520	6.10	1.20	34.48	160	2.60	0.50	64.00	36	9	0	84
c_15	20	50	3520	30.80	19.30	30.25	224	16.20	0.00	63.26	30	3	0	64
c_16	20	50	3520	34.40	2.00	40.97	542	1.90	0.90	13.41	5	23	2	43
c_17	20	50	3520	119.30	24.50	46.28	728	24.70	0.00	26.25	7	15	0	50
c_18	20	50	3520	50.10	10.90	40.22	661	11.90	10.60	24.42	20	3	2	132
c_19	20	50	3520	28.60	0.00	33.00	105	0.20	0.00	19.03	11	30	0	85

TABLE A3 | Comparison of the number of nodes in the branch-and-cut tree and the total computational time between the classical formulation and extended formulation on CSAP-Dense datasets.

Name	Classical formulation		Extended formulation	
	Nodes	Time (s)	Nodes	Time (s)
a_0	2557	500.55	223	253.50
a_1	1769	358.78	0	143.50
a_2	123	169.31	0	156.60
a_3	102	130.68	22	230.10
a_4	5	90.43	0	138.30
a_5	47	133.85	0	106.10
a_6	75	129.99	4	188.30
a_7	43	89.75	0	92.00
a_8	7965	472.57	0	136.80
a_9	0	104.23	0	151.30
a_10	10 057	1323.54	16	254.20
a_11	26	112.44	2	165.60
a_12	104	106.57	4	163.70
a_13	23	70.39	0	117.60
a_14	2624	205.82	0	194.30
a_15	258	122.26	7	134.25
a_16	185	95.15	119	189.52
a_17	22	81.12	1	101.44
a_18	0	80.46	0	72.48
a_19	20 784	620.26	17	187.11
b_0	945	132.38	103	155.10
b_1	106	96.97	3	131.10

(Continues)

TABLE A3 | (Continued)

Name	Classical formulation		Extended formulation	
	Nodes	Time (s)	Nodes	Time (s)
b_2	35	108.21	0	84.20
b_3	2039	294.21	0	84.20
b_4	0	44.87	0	129.70
b_5	620	117.86	34	113.10
b_6	95	107.04	4	97.70
b_7	528	144.84	175	112.50
b_8	8	49.68	5	64.60
b_9	73	89.98	4	60.90
b_10	1806	271.34	227	147.00
b_11	137	108.48	30	165.20
b_12	3	67.07	0	83.50
b_13	124	86.15	10	98.60
b_14	2746	309.11	0	80.40
b_15	36	54.13	6	107.90
b_16	0	37.13	0	20.30
b_17	351	130.94	9	89.30
b_18	60 585	TL	29	79.10
b_19	259	120.36	0	111.20
c_0	79	54.17	397	94.60
c_1	124 355	TL	315	84.00
c_2	40 703	1154.86	80	67.70
c_3	158 427	TL	0	24.90
c_4	0	27.34	0	16.80
c_5	116 004	TL	0	33.90
c_6	8360	168.32	0	15.20
c_7	58 762	947.99	63	30.00
c_8	0	38.29	3	22.20
c_9	167 405	TL	121	57.70
c_10	0	39.26	0	41.30
c_11	0	31.71	0	13.90
c_12	143 208	TL	19	38.10
c_13	85 109	TL	310	96.60
c_14	291	80.74	15	91.40
c_15	1492	115.05	21	80.44
c_16	200 972	TL	36	26.14
c_17	107 065	TL	0	26.21
c_18	139 595	TL	583	96.42
c_19	3	38.23	0	19.27

TABLE A4 | Root-node bounds obtained with the extended formulation on CSAP-Dense (without our cut procedures), along with branch-and-bound nodes, CPLEX cuts added and total computational time.

Problems			Before cutting		After cutting				
Name	S	T	Edges	Gap (%)	Gap (%)	Time (s)	#CPX	Nodes	Time(s)
a_0	35	35	2485	2.10	0.40	196.70	62	136	238.19
a_1	35	35	2485	1.80	0.00	196.50	49	0	194.86
a_2	35	35	2485	2.20	0.00	139.16	60	0	139.40
a_3	35	35	2485	1.90	0.40	242.27	63	29	265.41
a_4	35	35	2485	1.60	0.00	135.43	64	0	138.73
a_5	35	35	2485	1.10	0.00	123.52	53	0	127.46
a_6	35	35	2485	1.70	0.10	192.57	54	1	195.39
a_7	35	35	2485	0.80	0.00	108.72	39	0	111.43
a_8	35	35	2485	1.50	0.00	190.18	38	2	203.47
a_9	35	35	2485	1.80	0.00	148.81	54	0	154.73
a_10	35	35	2485	2.60	0.70	239.81	145	3576	415.36
a_11	35	35	2485	1.30	0.10	218.44	35	0	226.23
a_12	35	35	2485	2.30	0.10	171.75	64	1	175.21
a_13	35	35	2485	1.00	0.10	126.85	45	0	131.56
a_14	35	35	2485	1.70	0.00	186.97	38	0	188.63
a_15	35	35	2485	2.40	0.20	107.61	54	86	138.05
a_16	35	35	2485	2.20	0.70	161.34	60	117	220.21
a_17	35	35	2485	0.60	0.20	79.59	51	21	100.79
a_18	35	35	2485	1.60	0.00	81.21	42	0	80.49
a_19	35	35	2485	2.50	0.40	161.87	65	14	188.85
b_0	25	45	3175	2.20	0.20	122.70	94	139	161.83
b_1	25	45	3175	1.90	0.30	145.13	57	17	159.64
b_2	25	45	3175	1.30	0.00	114.21	78	0	107.11
b_3	25	45	3175	1.10	0.30	109.85	57	44	124.16
b_4	25	45	3175	2.00	0.00	147.30	66	0	147.20
b_5	25	45	3175	1.40	0.20	118.08	87	40	137.65
b_6	25	45	3175	0.90	0.00	115.26	78	6	118.57
b_7	25	45	3175	1.40	0.40	92.01	122	309	129.88
b_8	25	45	3175	0.60	0.10	73.99	48	3	79.72
b_9	25	45	3175	1.20	0.00	91.32	57	4	93.49
b_10	25	45	3175	1.30	0.30	118.21	142	721	179.82
b_11	25	45	3175	1.30	0.10	144.05	68	20	161.53
b_12	25	45	3175	1.60	0.00	109.33	79	0	112.81
b_13	25	45	3175	1.60	0.10	126.07	62	14	134.70
b_14	25	45	3175	1.20	0.00	101.78	30	0	101.18
b_15	25	45	3175	1.20	0.10	84.46	80	6	89.77
b_16	25	45	3175	0.30	0.00	31.45	38	0	33.15
b_17	25	45	3175	2.10	0.10	110.95	64	15	120.18
b_18	25	45	3175	1.50	0.80	59.85	55	53	86.26
b_19	25	45	3175	1.50	0.20	142.31	80	69	158.40
c_0	20	50	3520	17.30	3.60	36.17	125	209	88.15
c_1	20	50	3520	37.30	22.80	32.74	89	73	69.08
c_2	20	50	3520	20.30	11.40	43.90	255	1760	160.19
c_3	20	50	3520	8.40	0.00	19.96	55	0	19.49
c_4	20	50	3520	19.10	0.00	30.35	71	0	30.09
c_5	20	50	3520	2.50	1.00	27.76	60	4	31.80
c_6	20	50	3520	33.50	0.00	23.68	30	0	24.74
c_7	20	50	3520	6.10	5.80	19.79	58	36	31.68
c_8	20	50	3520	10.90	5.80	20.78	52	4	25.01

(Continues)

TABLE A4 | (Continued)

Problems			Before cutting		After cutting				
Name	S	T	Edges	Gap (%)	Gap (%)	Time (s)	#CPX	Nodes	Time(s)
c_9	20	50	3520	21.50	18.10	23.65	142	418	75.99
c_10	20	50	3520	0.60	0.00	70.52	85	0	68.51
c_11	20	50	3520	0.20	0.00	26.12	54	0	24.07
c_12	20	50	3520	7.70	0.00	32.22	68	0	36.26
c_13	20	50	3520	11.70	10.80	32.08	502	8572	292.62
c_14	20	50	3520	2.60	0.60	63.42	106	29	88.20
c_15	20	50	3520	16.20	0.10	54.10	80	112	90.96
c_16	20	50	3520	1.90	0.90	15.02	46	9	20.10
c_17	20	50	3520	24.70	0.00	22.49	44	2	25.23
c_18	20	50	3520	11.90	10.74	23.15	124	294	79.09
c_19	20	50	3520	0.20	0.00	20.58	74	1	23.72