A Branch-and-Price-and-Cut Algorithm for Discrete Network Design Problems Under Traffic Equilibrium

David Rey[1] and Michael W. Levin[2]

[1]SKEMA Business School, Université Côte d'Azur, Sophia Antipolis, France, david.rey@skema.edu
[2]Department of Civil, Environmental, and Geo- Engineering, University of Minnesota, Minneapolis, USA, mlevin@umn.edu

November 21, 2024

Abstract

This study addresses discrete network design problems under traffic equilibrium conditions or DNDPs. Given a network and a budget, DNDPs aim to model all-or-nothing decisions such as link addition to minimize network congestion effects. Congestion is measured using traffic equilibrium theory where link travel times are modeled as convex flow-dependent functions and where users make selfish routing decisions. In this context, the collective route choice of users is a Wardropian equilibrium and DNDPs admit a bilevel optimization formulation where the leader represents the network designer, and the follower is a parameterized traffic assignment problem (TAP). This study introduces a novel and exact branch-and-price-and-cut (BPC) algorithm for DNDPs that exploits the structure of the problem and harnesses the potential of path-based formulations for column generation (CG). Leveraging the convexity and the separability of the objective function, we develop successive relaxations of the bilevel problem that lead to an efficient outer approximation scheme that relies on solving a sequence of linear programs. We combine this OA procedure with a CG approach whose pricing subproblem can be solved in polynomial-time. This scheme is embedded within a single tree BPC algorithm to determine lower bounds while upper bounds are computed by solving parameterized TAPs. Numerical experiments conducted on a range of DNDP instances based on three transportation networks reveal that our BPC algorithm significantly outperforms state-of-the-art methods for DNDPs. Notably, we close several open instances of the literature, and we show that our BPC algorithm can solve DNDP instances based on networks whose number of nodes and of commodities is one order of magnitude larger than any previously solved instance.

Keywords: Bilevel optimization, Branch-and-price-and-cut, Network design, Traffic equilibrium, Selfish routing

1 Introduction

We consider discrete network design problems under traffic equilibrium. Discrete network design problems aim to model all-or-nothing decision problems, such as link or node addition, in a network. Traffic equilibrium refers to selfish routing and congestion effects in a network. In discrete network design problems under traffic equilibrium or DNDPs for short, the goal is to minimize congestion effects while accounting for users' collective route choice in response to the design. Such problems have well-documented applications in transportation and routing (Magnanti and Wong 1984) and can be used to inform decision-making in problems where the collective behavior of users influences congestion and therefore the choice of the network design. It-is well-known that DNDPs are NP-hard even if link travel time functions—also known as latency functions or delay functions—are linear (Roughgarden 2006). In transportation planning applications, multiple commodities representing the travel of users between their origin and destination often coexist and congestion functions are typically modeled as convex nonlinear functions of link flow (Colson et al. 2007). DNDPs are often modeled using bilevel optimization formulations where the leader represents the network designer and the follower represents a parameterized traffic equilibrium problem (Yang and H. Bell 1998). In this context, identifying the network design that optimizes the designer's objective function is notoriously challenging. Existing methods to solve DNDPs exactly do not scale well, mainly due to their inability to optimize over large networks. This difficulty stems from the nonlinearity of link travel time functions but also from the necessity to compute network equilibrium flows repeatedly to find optimal solutions.

The contributions of this study are as follows: we develop a novel methodology to solve DNDPs to optimality at scale. Unlike most existing approaches to solve DNDPs, we formulate a path-based convex relaxation of the original bilevel optimization problem. We combine outer approximation (OA) (Duran and Grossmann 1986, Fletcher and Leyffer 1994) and column generation (CG) (Dantzig and Wolfe 1960, Lübbecke and Desrosiers 2005) to obtain a tractable linear model that efficiently computes lower bounds. We propose a single tree branch-and-cut-and-price (BPC) algorithm that gradually adds cuts and path variables to this linear model and exploits the performance of state-of-the-art traffic assignment algorithms to computer upper bounds. We enrich our BPC algorithm with initialization heuristics and new value function cuts. Extensive numerical experiments that compare the performance of our BPC algorithm with three benchmarks, including two exact algorithms from the literature, highlight the benefits of the proposed approach. Notably, these experiments show that the solving the linear path-based model via a CG approach is computationally efficient and avoids scaling issues from solving a link-based multicommodity network design problem. Our numerical results solve a number of unsolved DNDP instances and we introduce new larger problem instances on medium to large-sized transportation networks to the community. For reproducibility purposes, all data and codes of this study are made publicly available.

The remainder of this paper is organized as follows. In Section 2, we review prior work on network design, selfish routing and traffic equilibrium. We present the problem formulation in Section 3. The methodological components of our approach including relaxations, OA and CG procedures are presented in Section 4. Our BPC algorithm is introduced in Section 5. Numerical experiments are reported in Section 6. A summary of our findings and research perspectives are provided in Section 7.

2 Literature Review

We first review studies on selfish routing and traffic equilibrium in Section 2.1 before discussing network design problems in Section 2.2. We emphasize our contributions relative to the state-of-the-art in Section 2.3.

2.1 Selfish routing and traffic equilibrium

A milestone in the study of traffic equilibrium is the seminal work of Wardrop (1952) who introduced the principle of traffic equilibrium to characterize the Nash equilibrium resulting from users' selfish routing decisions in a network: if users seek to minimize their own travel times, i.e. behave selfishly, their collective behavior is a Wardrop equilibrium if no user may reduce her travel time by unilaterally switching routes. Beckmann et al. (1956) later proposed a convex optimization formulation for the traffic assignment problem (TAP) to compute user equilibrium (UE) and system optimum (SO) network flows. The impact of selfish routing onto network congestion effects received an increased attention after the Braess (1968) paradox showed that adding capacity onto a network may worsen traffic conditions, and the worst-case price of anarchy was quantified by Roughgarden and Tardos (2002) and Correa et al. (2004). These results highlight that optimizing the design of a network under traffic equilibrium is nontrivial. Several variants of the TAP have been subsequently investigated to study richer traffic equilibrium models (Dafermos 1980), including bounded rationality (Mahmassani and Chang 1987) and stochastic user equilibrium (Dial 1971). Nevertheless, the original TAP is sufficiently challenging to motivate recent work on improved solution algorithms (Bar-Gera 2010, Xie and Xie 2016) and therefore remains relevant to the far more difficult problem of network design under traffic equilibrium.

2.2 Network design problems

Network design problems (NDPs) can be separated into continuous and discrete network design problems based on the type of decision modeled (Magnanti and Wong 1984). Continuous network design problems allow decisions to take real values whereas discrete problems focus on discrete decisions. Both continuous and discrete NDPs under traffic equilibrium have been studied in the literature. The problem of adding continuous link capacity to minimize network-wide congestion under traffic equilibrium is known as the continuous network design problem (CNDP) (Abdulaal and LeBlanc 1979). Since this study is concerned with discrete NDPs, we omit studies on CNDPs in this review of the literature and refer interested readers to Gairing et al. (2017). In addition, in the remainder of this section, we focus our attention to exact approaches for solving discrete network design problems under traffic equilibrium.

Most solution methods for DNDPs are based on the so-called System-Optimum (SO) relaxation which corresponds to the high-point relaxation of the bilevel optimization formulation (Colson et al. 2007). The first exact method for the link addition DNDP was introduced by Leblanc (1975), who combined the SO relaxation of the DNDP with a customized branching scheme to obtain valid lower bounds via the solution of SO-TAPs. The principle of Leblanc (1975)'s algorithm consists of using labels for tracking the status of candidate links. All candidate links are initially labeled as *unfixed*. At each node of the branch-and-bound (BB) tree, the SO-TAP with all *unfixed* links opened is solved to obtain a lower bound at this node. Since Braess' paradox effects do not occur if traffic is assigned under the SO principle, this provides valid lower bounds. Branching is performed on *unfixed* links and upper bounds are obtained by solving UE-TAPs upon reaching budget-feasible nodes. Since Leblanc (1975)'s algorithm exclusively relies on solving SO- and UE-TAPs, it can directly benefit from recent algorithmic advances for solving TAPs (Bar-Gera 2010, Xie and Xie 2016).

Gao et al. (2005) used generalized Benders' decomposition to develop an exact algorithm for the link addition DNDP. In this Benders' decomposition approach, the master problem is a knapsack problem enriched with Benders' optimality cuts while the subproblem is a parameterized TAP. The computation of Benders' optimality cuts relied on the determination of optimal dual variables of the Benders' subproblem which is nontrivial since the TAP is a convex nonlinear program. The authors introduce a single commodity instance with 20 nodes, 17 links and 6 variable links to test

their approach and also use an instance based on SiouxFalls network which consists of 24 nodes, 76 links, and 5 variable links. Farvaresh and Sepehri (2013) showed that Gao et al. (2005)'s derivation of Benders' optimality cuts was erroneous since these cuts were derived based on UE-TAP solutions instead of SO-TAPs ones. Farvaresh and Sepehri (2013) were the first to exploit the convexity of the objective function of DNDPs and use OA methods to obtain a relaxed mixed-integer linear model for the link addition SO-DNDP to determine improved lower bounds. Using the same branching scheme as Leblanc (1975), they employ Fletcher and Leyffer (1994)'s OA algorithm to solve the SO-DNDP at each node of the BB tree. This is shown to produce tighter lower bounds compared to Leblanc (1975)'s algorithm but requires the solution of multiple mixed-integer linear programs (MILPs) to solve SO-DNDPs. They solve DNDP instances on an extended SiouxFalls network with 100 nodes and 317 links, for a problem instance with 15 variable links.

Wang et al. (2013) considered a DNDP where discrete levels of link capacity may be added to existing links. Two methods are proposed to solve this DNDP: a first relaxation of the bilevel problem into a single level, convex mixed-integer nonlinear programming (MINLP) is proposed to get lower bounds while upper bounds are computed by solving best-response TAPs. Interdiction cuts are added to forbid the last design vector to be repeated, thus producing a cut generation algorithm that converges to the optimal bilevel solution by alternatively solving convex MINLPs and parameterized UE-TAPs. This method is then reinforced with value function cuts that aim to cut out bilevel-infeasible solution in the relaxed model. A mixed-integer OA model is used to represent nonlinear link travel time functions and numerical experiments on SiouxFalls network with 5 links available for capacity expansion and two levels of expansion are considered. Bagloee et al. (2017) proposed a BB algorithm which uses a generalized Benders' decomposition approach to solve the SO-relaxation of a multimodal DNDP at each node of the tree. The authors introduce a parameter to adjust the UE-SO gap which affect the validity of the obtained lower bounds, thus optimal solutions may be not guaranteed if this parameter is used. Results on SiouxFalls and Winninpeg's network are reported with up to 20 candidate links.

Mixed, i.e. discrete-continuous, NDPs have also received some attention. Luathep et al. (2011) study a mixed discrete-continuous NDP where both new links and the capacity expansion of existing links are modeled. They developed an SO-relaxation based approach where follower optimality conditions, i.e. UE, is ensure by the gradual addition of variational inequalities. The authors report numerical results for Gao et al. (2005)'s instance, and for instances based on SiouxFalls network with 5 and 10 candidate projects for pure discrete and mixed discrete-continuous instances. Wang et al. (2015) addressed a different mixed discrete-continuous NDP where the network designer aims to decide which links to add and their capacity. The authors used the variational inequality formulation of Luathep et al. (2011) but focused on developing an OA model of link travel time functions. Numerical results are only reported for Gao et al. (2005)'s instance.

It is also noted that some studies used piecewise-linear approximations of link travel time functions to create an approximate problem known as the linearized link addition DNDP which is then solved exactly (Farvaresh and Sepehri 2011, Fontaine and Minner 2014, Rey 2020). Farvaresh and Sepehri (2011) proposed a single level MILP formulation where follower optimality is enforced via linearized UE conditions. Linearized DNDP instances with up 16 nodes, 17 links and 25 candidate links are solved. Fontaine and Minner (2014) exploits the linearity of the linearized DNDP to derive a single level reformulation of the bilevel problem where follower optimality is ensured by a strong duality constraint. This reformulation is then solved using Benders' decomposition and numerical results on linearized DNDP instances on SiouxFalls network with 15 candidate links and on BerlinMitteCenter network, which consists of 398 nodes and 871 links, with 10 candidate links are reported. Rey (2020) conducted a computational study on three exact algorithms for the linearized link addition DNDP which revealed that 20-candidate link problem instances remained challenging to solve, even using

piecewise linear approximations of link travel time functions.

2.3 Our contributions

Our objective is to solve discrete network design problems under traffic equilibrium exactly. Our methodology builds on the bilevel optimization formulation of DNDPs and we further devote our attention to nonlinear link travel time functions—as opposed to linear travel time functions—which are commonly used in transportation network design studies. The main contribution of this study is to propose a novel solution method for DNDPs that leverages the potential of path-based network design formulations to scale-up. Our approach exploits the properties of the leader objective function, i.e. convexity and separability, to design a tight OA model that is amenable to an efficient CG procedure. Notably, the pricing suproblem of the CG procedure consists of solving a series of commodity-based shortest path problems which can be executed in polynomial-time. This framework is used to design a branch-and-price-and-cut (BPC) algorithm that gradually refines the OA model used to determine lower bounds by cut and column generation. Upper bounds are determined by solving best-response parameterized TAPs during search. This leads to a novel single tree BPC algorithm that is enriched with initialization heuristics, and interdiction and value function cuts.

To demonstrate the performance of our BPC algorithm, we conduct an extensive numerical benchmarking on a total of 180 DNDP instances based on three transportation networks of varying sizes. The performance of our BPC algorithm is compared to its link-based counterpart that does not require path variables and uses commodity-based link flows instead. This leads to a branch-and-cut (BC) algorithm that is identical to the proposed BPC except that it does not require CG. In addition, we consider two benchmarks from the literature: Leblanc (1975)'s branch-and-bound (BB) algorithm, which introduced the labeling scheme that we adopted and only relies on the solution of UE- and SO-TAPs; and Farvaresh and Sepehri (2013)'s OA BB algorithm, which also uses the same labeling scheme and uses OA to determine lower bounds during search. The former approach benefits from its reliance on TAP algorithms: since the introduction of Leblanc (1975)'s BB algorithm about 50 years ago, enormous computational gains have been obtained by improving TAP algorithms and state-of-the-art methods have given a second life to this seminal algorithm. The latter is the algorithm in the literature that is the most similar to our BPC algorithm since they both exploit the convexity of the leader objective function to determine lower bounds via an OA model. The most stringent difference is that Farvaresh and Sepehri (2013) used a link-based model that cannot scale to larger networks. Another key difference is that their OA model and implementation do not exploit the separability of the leader objective function. The extensive numerical experiments conducted highlight the substantial benefits of the path-based model and its associated CG approach compared to link-based counterparts. We also observe substantial gains compared to Leblanc (1975)'s seminal BB algorithm, thus outlining the scale-up capabilities of our approach.

3 Problem Formulation

DNDPs can be formulated as bilevel optimization problems where the leader problem aims to identify the optimal selection of a discrete resource to add to a network to minimize the total system travel time (TSTT) and the follower problem represents users' reaction, typically as a static traffic assignment problem (TAP) under Wardrop's user equilibrium (UE) principle (Wardrop 1952). For presentation and experimentation purposes, we use the link addition DNDP—hereby referred to as DNDP—as our target discrete network design problem under traffic equilibrium since this is the most studied DNDP in the literature.

The DNDP can be defined on a network with a node set $\mathcal{N}$ and link set $\mathcal{A}$ as a multicommodity network flow problem with nonlinear link travel time functions. Let d_w be the demand for commodity $w \in \mathcal{W} \subset \mathcal{N} \times \mathcal{N}$. Let Π_w be the set of paths connecting commodity $w \in \mathcal{W}$ and let $\Pi = \cup_{w \in W} \Pi_w$ be the set of all paths. Let $[\delta_a^\pi]_{a \in \mathcal{A}, \pi \in \Pi}$ be the link-path incidence matrix of the network. We denote h_π the flow on path $\pi \in \Pi$ and $\boldsymbol{h} = [h_\pi]_{\pi \in \Pi}$ the vector of path flows. Let $t_a(\cdot)$ represent the travel time function on link $a \in \mathcal{A}$, typically modeled as a positive and increasing function of the total link flow x_a to ensure the uniqueness of the UE link flows. Let $\mathcal{A}_1$ be the set of existing links and let $\mathcal{A}_2$ be the set of candidate links to improve the network, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$. For each link $a \in \mathcal{A}_2$, let g_a be the cost of adding this link to the network and let $y_a \in \{0, 1\}$ be the variable representing this choice. Let B be the available budget for optimization.

The leader represents a network designer that aims to minimize the TSTT defined as the sum of $x_a t_a(x_a)$ over all links $a \in \mathcal{A}$, subject to a budget constraint capturing the cost of link addition decisions $\boldsymbol{y}$, hereby referred to as leader variables. The link flow pattern variables $\boldsymbol{x} = [x_a]_{a \in \mathcal{A}}$ are determined by the follower problem, which is the TAP formulation under UE (Beckmann et al. 1956, Leblanc 1975, Magnanti and Wong 1984). The impact of the leader variables $\boldsymbol{y}$ in the follower is achieved through the linking indicator constraints that require null flow on $\mathcal{A}_2$ links that remain closed.

The follower problem is a TAP based on Beckmann et al. (1956)'s formulation parameterized by leader variables $\boldsymbol{y}$:

$$\texttt{TAP}(\boldsymbol{y}) : \min_{\boldsymbol{x}, \boldsymbol{h}} \quad \sum_{a \in \mathcal{A}} \int_0^{x_a} t_a(v) dv \tag{1a}$$

$$\text{s.t.} \quad \sum_{\pi \in \Pi_w} h_\pi = d_w \qquad \forall w \in \mathcal{W} \tag{1b}$$

$$\sum_{\pi \in \Pi} h_\pi \delta_a^\pi = x_a \qquad \forall a \in \mathcal{A} \tag{1c}$$

$$x_a = 0 \quad \text{if } y_a = 0 \qquad \forall a \in \mathcal{A}_2 \tag{1d}$$

$$h_\pi \geq 0 \qquad \forall \pi \in \Pi \tag{1e}$$

Let $\texttt{TAP}(\boldsymbol{y})$ also denote the set of optimal link flow solutions $\boldsymbol{x}$ of the parameterized TAP (1). The DNDP can be formulated as the following bilevel optimization problem:

$$\min_{\boldsymbol{y}} \quad \sum_{a \in \mathcal{A}} x_a t_a(x_a) \tag{2a}$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{A}_2} y_a g_a \leq B \tag{2b}$$

$$y_a \in \{0, 1\} \qquad \forall a \in \mathcal{A}_2 \tag{2c}$$

$$\boldsymbol{x} \in \texttt{TAP}(\boldsymbol{y}) \tag{2d}$$

Note that although the follower problem contains two groups of variables: path flows and link flows, only the latter are used to compute the leader objective function. Furthermore, it is well-known that, if link travel time functions $t_a(\cdot)$ are positive and increasing then, for any leader decision $\boldsymbol{y}$, the objective function of the follower is strictly convex with regards to variables $\boldsymbol{x}$ and there is a unique UE link flow pattern $\boldsymbol{x} \in \texttt{TAP}(\boldsymbol{y})$ (Sheffi 1985).

4 Relaxations, Outer Approximation and Column Generation

In this section, we present the techniques used to obtain relaxed and tractable formulations of the DNDP. We start by relaxing the bilevel optimization problem into a single level problem via its System Optimum (SO) relaxation. We exploit the properties of the objective function of DNDP and propose a novel link-based Outer Approximation (OA) model to obtain a Mixed-Integer Linear Programming (MILP) relaxation of the SO relaxation. Leveraging the path-based formulation of this MILP relaxation, we develop a Column Generation (CG) with a polynomial-time pricing subproblem.

For presentation purposes, we define the following sets: let $\mathcal{X}$ be the set of (unrestricted) feasible link flows:

$$\mathcal{X} \equiv \left\{ \boldsymbol{x} \in \mathbb{R}^{|\mathcal{A}|} : \sum_{\pi \in \Pi} h_\pi \delta_a^\pi = x_a,\ \forall a \in \mathcal{A},\ \sum_{\pi \in \Pi_w} h_\pi = d_w,\ \forall w \in W, h_\pi \geq 0,\ \forall \pi \in \Pi \right\} \tag{3}$$

Let $\mathcal{Y}$ be the set of feasible link addition decisions:

$$\mathcal{Y} \equiv \left\{ \boldsymbol{y} \in \{0,1\}^{|\mathcal{A}_2|} : \sum_{a \in \mathcal{A}_2} y_a g_a \leq B \right\} \tag{4}$$

4.1 System Optimum Relaxation

Let $Q = \sum_{w \in \mathcal{W}} d_w$ be the total demand. A mathematical programming formulation of the system optimum DNDP, denoted SO-DNDP is:

$$\min_{\boldsymbol{y} \in \mathcal{Y}, \boldsymbol{x} \in \mathcal{X}} \quad \sum_{a \in \mathcal{A}} x_a t_a(x_a) \tag{5a}$$

$$\text{s.t.} \quad x_a \leq y_a Q \qquad \forall a \in \mathcal{A}_2 \tag{5b}$$

Here the linking constraint (1d) is reformulated into a so-called big-M constraint (5b) which is linear and for which a finite bound is known. The SO relaxation (5) of the DNDP formulation (2) is a MINLP with a convex objective function, which motivates the use of dedicated algorithms, notably OA methods.

4.2 Outer Approximation

The nonlinear link travel time functions $t_a(\cdot)$ make problem (5) difficult to solve. It is well-known that the objective function (2a) of the DNDP (and of SO-DNDP) is convex, which we use to write an OA of the objective function (Duran and Grossmann 1986, Fletcher and Leyffer 1994). Furthermore, the objective function of DNDP (TSTT) is separable with regards to link flow variables $\boldsymbol{x}$ (Beckmann et al. 1956), which means that the OA can be made tighter by creating a link-based OA instead of a single OA of the entire objective function. We exploit these properties to build a link-based OA of the terms $x_a t_a(x_a)$ composing the objective function.

Formally, given a vector $\boldsymbol{x}^k \in \mathcal{X}$ of feasible link flows, the gradient of $x_a t_a(x_a)$ at $\boldsymbol{x}^k$ is:

$$x_a^k t_a(x_a^k) + \frac{d\left[x_a^k t_a(x_a^k)\right]}{dx_a^k} \times (x_a - x_a^k) = x_a^k t_a(x_a^k) + (x_a^k t_a'(x_a^k) + t_a(x_a^k)) \times (x_a - x_a^k) \tag{6a}$$

$$= x_a \left(x_a^k t_a'(x_a^k) + t_a(x_a^k)\right) - (x_a^k)^2 t_a'(x_a^k) \tag{6b}$$

$$= x_a \alpha_a^k + \beta_a^k \tag{6c}$$

where $\alpha_a^k \equiv x_a^k t'_a(x_a^k) + t_a(x_a^k)$ and $\beta_a^k = -(x_a^k)^2 t'_a(x_a^k)$ are constants. Because of convexity of the TSTT objective function, for any $\boldsymbol{x}^k \in \mathcal{X}$, $x_a\alpha_a^k + \beta_a^k$ is a linear under-estimator of $x_a t_a(x_a)$.

Let $\mu_a \geq 0$ be a real decision variable representing the contribution of link $a \in \mathcal{A}$ to the objective function (2a). Let $\mathcal{C}_a$ be the set of indices k corresponding to the link flows x_a^k at which OA is performed, i.e. $\mathcal{C}_a$ represents the set of OA cuts used to under-estimate the contribution of link $a \in \mathcal{A}$ to the objective function. Given a collection of index sets $[\mathcal{C}_a]_{a\in\mathcal{A}}$, the following formulation is a MILP relaxation of SO-DNDP:

$$\min_{\boldsymbol{y}\in\mathcal{Y},\boldsymbol{x}\in\mathcal{X},\boldsymbol{\mu}\geq\boldsymbol{0}} \quad \sum_{a\in\mathcal{A}} \mu_a \tag{7a}$$

$$\text{s.t.} \quad \mu_a \geq x_a\alpha_a^k + \beta_a^k \qquad \forall a \in \mathcal{A}, k \in \mathcal{C}_a \tag{7b}$$

$$x_a \leq y_a Q \qquad \forall a \in \mathcal{A}_2 \tag{7c}$$

$$y_a \in \{0,1\} \qquad \forall a \in \mathcal{A}_2 \tag{7d}$$

Solving Formulation (7) yields a lower bound (LB) on the optimal objective function value (OFV) of DNDP. In contrast to the DNDP literature, this formulation exploits the link-separability property of the objective function of the DNDP. Farvaresh and Sepehri (2013) developed a MILP relaxation of SO-DNDP based on the OA of the entire objective function as opposed to link-based OAs. Their MILP formulation was also formulated using a link-based multicommodity network flow model. Instead, Formulation (7) is path-based and contains an exponential number of path flow variables $\boldsymbol{h}$. We exploit the MILP structure of Formulation (7) and its amenability to CG techniques to avoid enumerating all paths within Formulation (7).

4.3 Column Generation

To solve Formulation (7), we consider its linear programming (LP) relaxation. Let $\bar{\Pi} \subset \Pi$ be a restricted set of paths and let $\bar{\Pi}_w \subset \Pi_w$ be the corresponding commodity-based restricted path sets. The restricted master problem (RMP) is:

$$\min_{\boldsymbol{y},\boldsymbol{x},\boldsymbol{h},\boldsymbol{\mu}} \quad \sum_{a\in\mathcal{A}} \mu_a \tag{8a}$$

$$\text{s.t.} \quad \mu_a \geq x_a\alpha_a^k + \beta_a^k \qquad \forall a \in \mathcal{A}, k \in \mathcal{C}_a \tag{8b}$$

$$\sum_{a\in\mathcal{A}_2} y_a g_a \leq B \tag{8c}$$

$$\sum_{\pi\in\bar{\Pi}_w} h_\pi = d_w \qquad \forall w \in \mathcal{W} \tag{8d}$$

$$\sum_{\pi\in\bar{\Pi}} h_\pi \delta_a^\pi = x_a \qquad \forall a \in \mathcal{A} \tag{8e}$$

$$x_a \leq y_a Q \qquad \forall a \in \mathcal{A}_2 \tag{8f}$$

$$0 \leq y_a \leq 1 \qquad \forall a \in \mathcal{A}_2 \tag{8g}$$

$$h_\pi \geq 0 \qquad \forall \pi \in \bar{\Pi} \tag{8h}$$

Since the objective of (SO-)DNDP is to minimize network-wide congestion, link and path flows are indirectly minimized. Hence, constraints (8d) and (8e) can be rewritten as inequalities to restrict

the sign of their dual variables, namely:

$$\sum_{\pi \in \Pi_w} h_\pi \geq d_w \qquad \forall w \in W \tag{9a}$$

$$\sum_{\pi \in \Pi} h_\pi \delta_a^\pi \leq x_a \qquad \forall a \in A \tag{9b}$$

We denote $\sigma_w \geq 0$ the dual variable of the demand constraint (9a) and $\zeta_a \geq 0$ the dual variable of the link flow constraint (9b). Given a commodity $w \in \mathcal{W}$ and a path $\pi \in \Pi_w$, the reduced cost of variable h_π, denoted c_π, is:

$$c_\pi = -\sigma_w + \sum_{a \in A} \delta_a^\pi \zeta_a \tag{10}$$

The reduced costs of path-flow variables $\boldsymbol{h}$ can be computed in polynomial-time by solving, for each commodity $w \in \mathcal{W}$, a shortest path problem in the directed network $(\mathcal{N}, \mathcal{A})$ with link costs given by the dual vector $\boldsymbol{\zeta} = [\zeta_a]_{a \in \mathcal{A}}$ and deducing σ_w from the shortest path length.

This CG approach is novel and provides a paradigm shift for solving DNDPs at scale: as shown in our numerical experiments in Section 6, the pricing of path flows variables is computationally inexpensive and the path-based formulation of the SO-relaxed OA model of DNDP is more efficient than its link commodity-based counterpart.

5 Branch-and-Price-and-Cut Algorithm

We present the branch-and-price-and-cut (BPC) algorithm developed to solve the DNDP to optimality. Our BPC algorithm implements a single tree that solves a sequence of LPs in the form of RMP (8). Columns and cuts are added dynamically to obtain LBs. UBs are computed by determining the follower's best-response to a given leader decision: specifically, given an integral vector $\boldsymbol{y} \in \mathcal{Y}$, solving $\texttt{TAP}(\boldsymbol{y})$ gives an UB on the OFV of DNDP. Our algorithm builds on the labeling scheme introduced by Leblanc (1975) and refined by Farvaresh and Sepehri (2013) but differs from these approaches by exploiting the relaxed path-based linear model (8) and its associated CG approach. Throughout this section, we use index k to denote a node of the BPC tree and its associated branch cuts.

5.1 Labeling Scheme

We adopt the labeling scheme introduced by Leblanc (1975) to track the status of candidate links in $\mathcal{A}_2$ during search. We define the sets of fixed links at iteration k as:

$$\mathcal{A}_2^0(k) \equiv \left\{a \in \mathcal{A}_2 : y_a^k = 0\right\} \tag{11a}$$

$$\mathcal{A}_2^1(k) \equiv \left\{a \in \mathcal{A}_2 : y_a^k = 1\right\} \tag{11b}$$

Note that $(\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k)) \subseteq \mathcal{A}_2$, in particular at the root node: $\mathcal{A}_2^0(0) = \mathcal{A}_2^1(0) = \emptyset$. At each node k of the BPC tree, the budget constraint is checked for violations using the following procedure named $\texttt{check}(k)$ is executed:

- if $\sum_{a \in \mathcal{A}_2^1(k)} g_a > B$ the budget is violated and the BPC node is labeled as `infeasible`;
- else if $|\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k)| = |\mathcal{A}_2|$, then all links are fixed and the BPC node is labeled as `fixed`;

- else if $\sum_{a\in\mathcal{A}_2^1(k)} g_a < \min\{g_a : a \in \mathcal{A}_2 \setminus (\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k))\}$, then the budget is not violated but there is not enough budget to open any of the unfixed links. In this case, set $y_a^k = 0$ for all $a \in \mathcal{A}_2 \setminus (\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k))$, and the BPC node is labeled as `fixed`;
- else: the BPC node is labeled as `unfixed`.

If the `check`(k) procedure results in the label `unfixed`, the selected BPC node is further processed and a LB is computed by solving RMP (8) along with branch cuts by CG. If the outcome of the `check`(k) procedure is `fixed`, `TAP`$(\boldsymbol{y}^k)$ is solved to obtain an UB on the OFV of DNDP and the tree is pruned at this node. Analogously, if the outcome of the `check`(k) procedure is `infeasible`, the tree is pruned at this node.

5.2 Branching and Node Selection Rules

If a BPC node is not pruned, the algorithm selects a link $a \in \mathcal{A}_2 \setminus \{\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k)\}$ for branching. Priority is given to links whose y_a^k value is fractional and if none exists, branching is performed on unfixed links. In both cases, we use the scoring function used in the literature (Leblanc 1975, Farvaresh and Sepehri 2013, Rey 2020) to sort candidate links for branching: links are sorted by decreasing `score`$(a) = x_a^k t_a(x_a^k)$ values which corresponds to their contribution to the leader objective function. We use the best-bound first search rule to select the next BPC node during search.

5.3 OA Cut Generation

Leveraging the separability of the objective function, the generation of OA cuts can be performed on a per-link basis and a threshold rule is used to control the density of link-based OA cuts. For each link $a \in \mathcal{A}$ and for each cut index $k \in \mathcal{C}_a$, we store the link flow x_a^k at which OA is performed. Before extending the index set $\mathcal{C}_a$, we scan stored cut indices and check if the absolute value difference of the candidate link flow with stored link flows is greater than a predefined threshold ϵ_{OA}. Formally, given a candidate link flow x_a, if there exists no index $k \in \mathcal{C}_a$ such that $|x_a - x_a^k| \leq \epsilon_{OA}$, then we extend $\mathcal{C}_a$ by adding an index corresponding to the candidate link flow x_a. This process is hereby referred to as the (leader) OA procedure. In our BPC implementation, we execute the OA procedure after each RMP solve, i.e. link-based OA cuts are generated by computing the linear-underestimators (6c) at points $\boldsymbol{x}^k \in \mathcal{X}$ that solve the RMP (8).

5.4 Interdiction Cuts

To avoid repeating an integer solution and solving the corresponding parameterized TAP, interdiction cuts can be added to the RMP. Let $\mathcal{F}$ be the set of indices corresponding to the parameterized TAPs solved. For each $f \in \mathcal{F}$, we denote $\boldsymbol{y}^f = [y_a^f]_{a\in\mathcal{A}_2}$ the link addition vector corresponding to `TAP`$(\boldsymbol{y}^f)$. The interdiction cuts are:

$$\sum_{a\in\mathcal{A}_2:y_a^f=0} y_a + \sum_{a\in\mathcal{A}_2:y_a^f=1} (1-y_a) \geq 1 \qquad \forall f \in \mathcal{F} \tag{12}$$

In our implementation of the BPC algorithm, set $\mathcal{F}$ is initially empty and this set is gradually extended by adding $\boldsymbol{y}$-vectors each time the follower best-response is computed.

5.5 Value Function Cuts

To tighten the feasible region of RMP (8), we consider including value function (VF) cuts. VF cuts aim to cut out HPR-feasible solutions that are bilevel-infeasible. Lozano and Smith (2017) proposed a cutting-plane algorithm for bilevel optimization problems with integer leader decisions which iteratively adds VF cuts to the HPR. We build on this approach and derive customized VF cuts for the link addition DNDP.

Proposition 1. *Let $\boldsymbol{x}^f \in \texttt{TAP}(\boldsymbol{y}^f)$ be the link flow vector corresponding to the follower best-response for a given $\boldsymbol{y}^f \in \mathcal{Y}$ vector. Given a vector of large enough values $[M_a]_{a\in\mathcal{A}_2}$, the set of constraints:*

$$\sum_{a\in\mathcal{A}} \int_0^{x_a} t_a(v)dv \leq \sum_{a\in\mathcal{A}} \int_0^{x_a^f} t_a(v)dv + \sum_{a\in A_2: x_a^f>0} M_a(1-y_a), \qquad \forall f \in \mathcal{F} \tag{13}$$

are value function cuts for the link addition DNDP.

Proof. Proof. From Proposition 1 of Lozano and Smith (2017), a solution $(\boldsymbol{y}, \boldsymbol{x})$ is bilevel-feasible for link addition DNDP if and only if

$$\sum_{a\in\mathcal{A}} \int_0^{x_a} t_a(v)dv \leq \sum_{a\in\mathcal{A}} \int_0^{x_a^f} t_a(v)dv \qquad \forall \boldsymbol{x}^f \in \texttt{TAP}(\boldsymbol{y}) \tag{14}$$

In the link addition DNDP, the dependency of a leader strategy $\boldsymbol{y}$ and the follower best-response $\boldsymbol{x}^f \in \texttt{TAP}(\boldsymbol{y})$ occurs through the linking constraint (1d) which is represented in integer-linear form (8f) in RMP (8). Observe that, given an integral vector $\boldsymbol{y}^f \in \mathcal{Y}$, opening a link which is closed in $\boldsymbol{y}^f$ only extends the feasible region of the follower problem. However, any link $a \in \mathcal{A}_2$ with nonzero best-response flow $x_a^f > 0$ must be opened, i.e. $y_a^f = 1$, and closing this link makes the flow $\boldsymbol{x}^f$ infeasible. Therefore, the follower objective value may increase if links are closed, which requires adding the "big M" term $M_a(1 - y_a)$ to deactivate the constraints if any link $a \in \mathcal{A}_2$ with nonzero best-response flow $x_a^f > 0$ is closed, i.e. $y_a = 0$. □ □

We note that Wang et al. (2013) proposed VF cuts for a DNDP where the leader controls discrete link capacity expansion variables. In this model, the set of network links is fixed but the capacity of certain links can be increased by discrete amounts. This modeling choice avoids the problem of "blocking constraints" discussed in Lozano and Smith (2017) but cannot be applied to the link addition DNDP.

To incorporate the VF cuts (13) in RMP (8), we exploit the convexity and the separability of its left-hand-side, i.e. the TAP objective function (1a), to determine an under-estimator via OA and valid upper bounds on M_a terms. Let $B_a(x_a) \equiv \int_0^{x_a} t_a(v)dv$.

Given a feasible flow $\boldsymbol{x}^k \in \mathcal{X}$, the gradient of $B_a(x_a)$ at $\boldsymbol{x}^k$ is:

$$\int_0^{x_a^k} t_a(v)dv + \frac{d\left[\int_0^{x_a^k} t_a(v)dv\right]}{dx_a^k} \times (x_a - x_a^k) = \int_0^{x_a^k} t_a(v)dv + \sum_{a\in A} t_a(x_a^k) \times (x_a - x_a^k) \tag{15a}$$

$$= x_a t_a(x_a^k) + \int_0^{x_a^k} t_a(v)dv - x_a^k t_a(x_a^k) \tag{15b}$$

$$= x_a \gamma_a^k + \phi_a^k \tag{15c}$$

where $\gamma_a^k \equiv t_a(x_a^k)$ and $\phi_a^k = \int_0^{x_a^k} t_a(v)dv - x_a^k t_a(x_a^k)$ are constants.

Since the TAP objective function (1a) is decreasing with $\boldsymbol{y}$, it holds that $\sum_{a\in\mathcal{A}} B_a(x_a) \leq \sum_{a\in\mathcal{A}} B_a(x_a^0) = M_a = M$ where $\boldsymbol{x}^0 \in \texttt{TAP}(\boldsymbol{y} = \boldsymbol{0})$. Let $\mu_a^B \geq 0$ be real variables used to capture the contribution of link $a \in \mathcal{A}$ in the follower objective function (1a). Let $\mathcal{C}_a^B$ be the set of indices k corresponding to the link flows x_a^k at which OA is performed for the follower objective function (1a). The set of constraints:

$$\mu_a^B \geq x_a \gamma_a^k + \phi_a^k \qquad \forall a \in \mathcal{A}, k \in \mathcal{C}_a^B \tag{16a}$$

$$\sum_{a\in\mathcal{A}} \mu_a^B \leq \sum_{a\in\mathcal{A}} B_a(x_a^f) + M \sum_{a\in A_2: x_a^f > 0} (1 - y_a), \qquad \forall f \in \mathcal{F} \tag{16b}$$

represent a relaxation of the VF cuts (13) and are thus valid inequalities for the link addition DNDP.

5.6 Columns, OA cuts and UB Initialization

We initialize the restricted path sets $\bar{\Pi}_w$, for each commodity $w \in \mathcal{W}$, by closing all links in $\mathcal{A}_2$ and solving a shortest path problem using free-flow link travel times. This ensures that the initial restricted path set $\bar{\Pi} = \cup_{w\in\mathcal{W}} \bar{\Pi}_w$ is feasible for any link addition vector $\boldsymbol{y} \in \mathcal{Y}$.

To attempt to obtain a competitive root node LB, OA cuts index sets $[\mathcal{C}_a]_{a\in\mathcal{A}}$ are initialized by solving SO-TAPs with a relaxed optimality gap for a series of $\boldsymbol{y}$-vectors. We consider two heuristics to initialize OA cuts index sets. Both heuristics start by solving SO-TAP with $\boldsymbol{y} = \boldsymbol{1}$ and executing the leader OA procedure at the obtained link flow vector $\boldsymbol{x}^{\boldsymbol{1}}$. This aims to ensure that all links of the network have at least one OA cut. Subsequently, the binary knapsack problem $\max_{y\in\mathcal{Y}} \sum_{a\in\mathcal{A}_2} y_a x_a^{\boldsymbol{1}}$ is solved to obtain a feasible $\bar{\boldsymbol{y}}$, SO-TAP is solved for $\bar{\boldsymbol{y}}$ and the OA procedure is executed. In addition, the parameterized follower $\texttt{TAP}(\bar{\boldsymbol{y}})$ is solved to obtain an UB on the OFV of DNDP. Then, one of two following heuristics is used:

- `kBestKP`: Add an interdiction cut of the form (12) to the binary knapsack problem, obtain a "second-best" $\boldsymbol{y}$ vector, solve SO-TAP and execute the OA procedure, etc; for a predefined number of iterations.
- `LocalSearchKP`: Explore the neighborhood of $\bar{\boldsymbol{y}}$ by turning on or off each link one at a time, solve SO-TAP at this neighbor and execute the OA procedure.

5.7 Algorithm Overview

The pseudo-code of the BPC algorithm is presented in Algorithm 1. The algorithm starts by initializing the (global) lower and upper bounds LB and UB, the restricted path set $\bar{\Pi}$ and link-based OA cuts index sets $[\mathcal{C}_a]_{a\in\mathcal{A}}$. At each BPC iteration, three boolean control variables `prune`, `runOA`, `runTAP` are initialized to `False`: `prune` controls tree pruning; while `runOA` and `runTAP` determine whether to update the link-based OA models by attempting to add new OA cuts and to compute the leader OFV by solving a parameterized (UE-)TAP to attempt to update UB, respectively.

After selecting an active BPC node k, the procedure $\texttt{check}(k)$ is executed: if it returns `infeasible`, then the boolean `prune` is set to `True`. If it returns `fixed`, both `prune` and `runTAP` are set to `True` and the leader OFV is computed at the corresponding $\boldsymbol{y}^k$ vector. If the procedure $\texttt{check}(k)$ returns `unfixed`, the RMP (8) at the current BB node is solved by CG: the BPC node is pruned if the CG procedure is infeasible or if the local LB greater than the OFV of the best known bilevel-feasible solution, i.e. $LB_k \geq UB$. Otherwise, the boolean `runOA` is set to `True`. If the BPC node is not pruned, the branching rule is used to find the branching link $a \in \mathcal{A}_2 \setminus \{\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k)\}$ with the largest $\texttt{score}(a)$ value: priority is to branch on fractional variables, and if none exists, on unfixed

variables. In the latter case, one of the children BPC node needs not to be solved again and Line 13 can be skipped. If the node is pruned or after branching, the current node is labeled as inactive. The algorithm terminates if the relative gap between LB and UB falls within a predetermined threshold or if there are no more active BPC nodes.

We establish the correctness and the finiteness of our BPC algorithm in the following proposition.

Proposition 2. *The BPC algorithm 1 solves the DNDP to optimality in a finite number of iterations.*

Proof. Proof. Finiteness stems from the discreteness of the DNDP: since there is a finite number of link addition variables, the maximum number of BB iterations is $2^{|\mathcal{A}_2|}$. To demonstrate correctness, i.e. that the algorithm solves the DNDP exactly, due to Braess paradox effects, we must show that no integral $\boldsymbol{y}$ vector capable of improving the best known bilevel-feasible solution is skipped during search, and that upon termination the algorithm has proven this best known bilevel-feasible solution to be a global optimum.

Since RMP (8) is a relaxation of SO-DNDP (5) which is itself a convex relaxation of the bilevel optimization problem (2), solving RMP (8) by CG at the root node of the BPC tree yields a LB on the optimal OFV of (2). This LB is gradually refined through branching and at the end of any BPC iteration, LB is updated by taking the minimum over LBs of active BPC nodes using standard BB logic. Throughout search, UB is determined by solving $\boldsymbol{y}$-parameterized (UE-)TAPs at integral $\boldsymbol{y}$ points and is updated whenever the leader OFV computed at the follower best-response link flow point $\boldsymbol{x}$ improves on the incumbent UB. Hence, at any BPC iteration LB and UB are valid lower and upper bounds on the optimal OFV of the bilevel optimization problem (2).

To see that no integral $\boldsymbol{y}$ vector capable of improving the best-known bilevel-feasible solution has been skipped during search, observe that if the `check`(k) procedure returns `fixed`, the leader OFV is computed by solving the $\boldsymbol{y}^k$-parameterized (UE-)TAP and this branch of the tree can be fathomed since all variable links have been fixed. If the `check`(k) procedure returns `infeasible`, the branch can be fathomed since the budget constraint is violated. Otherwise, at least one link is unfixed, and a local LB is determined by solving RMP (8) by CG at k: this BPC node is pruned only if the RMP is infeasible or if the local LB exceeds the best-known bilevel feasible solution. □ □

The implementation of Algorithm 1 can be tuned without affecting finitess or correctness by adding interdiction cuts and value function cuts after solving the parameterized UE-TAP at an integral point $\boldsymbol{y}^k$ as indicated in the optional step at Line 25.

6 Numerical Results

We start by presenting the design of our numerical experiments to validate the performance of our BPC algorithm.

6.1 Experiments Design

We conduct numerical experiments to test the performance of the BPC algorithm for the link addition DNDP. We consider three benchmarks: a branch-and-cut (BC) algorithm which follows the same single tree BB algorithm as BPC but uses a commodity link-based formulation instead of the path-based formulation, therefore no CG is required. Therefore, the only difference between BPC and BC is the use of a path-based formulation and the CG procedure. We also compare BPC and BC with two benchmark algorithms from the literature: Leblanc (1975)'s BB algorithm—hereby referred to as Leblanc—which solves SO-TAPs to determine LBs; and Farvaresh and Sepehri (2013)'s BB algorithm which uses Fletcher and Leyffer (1994)'s OA algorithm to determine LBs—hereby

Algorithm 1: Branch-and-Price-and-Cut (BPC) algorithm for the DNDP

$\bar{\Pi} \leftarrow$ initialize path variables
$LB \leftarrow 0$
$UB, [\mathcal{C}_a]_{a\in\mathcal{A}} \leftarrow$ execute heuristic to initialize OA cuts and compute initial UB
Initialize BB tree with root node 0
while *not converged* **do**
  $k \leftarrow$ select an active BB node
  `prune`, `runOA`, `runTAP` $\leftarrow$ `False`
  if `check`$(k) =$ `infeasible` **then**
    `prune` $\leftarrow$ `True`
  else if `check`$(k) =$ `fixed` **then**
    `prune`, `runTAP` $\leftarrow$ `True`
  else
    $LB_k, \boldsymbol{y}^k, \boldsymbol{x}^k \leftarrow$ solve RMP (8) at node k by CG
    if *RMP is infeasible or if* $LB_k \geq UB$ **then**
      `prune` $\leftarrow$ `True`
    else
      `runOA` $\leftarrow$ `True`
  if `runOA` $=$ `True` **then**
    Update $[\mathcal{C}_a]_{a\in\mathcal{A}}$ at $\boldsymbol{x}^k$
  if `runTAP` $=$ `True` **then**
    $UB_k \leftarrow$ solve UE-TAP at $\boldsymbol{y}^k$
    $UB \leftarrow \min\{UB, UB_k\}$
    Optional: add interdiction cut and value function cut at $\boldsymbol{y}^k$
  if `prune` $=$ `False` **then**
    $\mathcal{A}_2^{\texttt{frac}}(k) \leftarrow \{a \in \mathcal{A}_2 \setminus \{\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k)\} : y_a^k \text{ is fractional}\}$
    if $|\mathcal{A}_2^{\texttt{frac}}(k)| > 0$ **then**
      $a \leftarrow \arg\max\{\texttt{score}(a) : a \in \mathcal{A}_2^{\texttt{frac}}(k)\}$
    else
      $a \leftarrow \arg\max\{\texttt{score}(a) : a \in \mathcal{A}_2 \setminus \{\mathcal{A}_2^0(k) \cup \mathcal{A}_2^1(k)\}\}$
    Create two children nodes with $y_a^{k+1} = 0$ and $y_a^{k+2} = 1$
  Update LB based on active BB nodes and check convergence

Network (Acronym)	**Nodes**	**Links**	**Commodities**	**Trip inflation factor**
SiouxFalls (SF)	24	76	528	1
Eastern Massachusetts (EM)	74	258	1113	4
BerlinMitteCenter (BMC)	398	871	1260	2

Table 1: Transportation networks used for generating DNDP instances.

referred to as FS_NETS. All four implemented algorithms, i.e. BPC, BC, FS_NETS and Leblanc and; use the same `check`(k) procedure to scan a BB node k before further processing it if it is labeled `unfixed`. All TAPs are solved using our implementation of Bar-Gera (2010)'s TAPAS algorithm.

We use three transportation networks from a public repository containing traffic assignment, i.e. network and trips, data (Transportation Networks for Research Core Team 2024) to generate DNDP instances: SiouxFalls (SF), Eastern Massachusetts (EM) and BerlinMitteCenter (BMC). SF is a test network with 24 nodes, 76 links and 528 commodities widely used in DNDP studies (Farvaresh and Sepehri 2013, Wang et al. 2013, Rey 2020). EM contains 74 nodes, 258 links and 1113 commodities; and BMC contains 398, 871 links and 1260 commodities—the latter network was also used by Fontaine and Minner (2014) for solving the linearized link addition DNDP. To increase congestion effects on EM and BMC networks, we inflate trips by a factor of 4 and 2, respectively. Network information is summarized in Table 1.

We use the SF DNDP instances introduced by Rey (2020) which consist of 20 instances: 10 with 10 additional new links, i.e. $|\mathcal{A}_2| = 10$ thus $|\mathcal{A}| = 86$; and 10 with 20 additional new links, i.e. $|\mathcal{A}_2| = 20$ thus $|\mathcal{A}| = 96$. For the EM and BMC networks, since these networks already have many links, we generate DNDP instances by randomly sampling $|\mathcal{A}_2|$ links among existing links. For each sample, we verify the impact of closing these links one at a time by solving the corresponding UE-TAP and recording the TSTT percentage change relative to the original network. We discard samples if more than $1/3$ of the sampled links do not generate an absolute change in TSTT greater than 1%. For both EM and BMC networks, we generate 20 DNDP instances: 10 with $|\mathcal{A}_2| = 10$ and 10 with $|\mathcal{A}_2| = 20$ and the cost of opening $\mathcal{A}_2$ is determined by randomly perturbing a linear function of links' free-flow travel time and capacity. For all three networks and for each of the 20 samples of $\mathcal{A}_2$ links and their costs, we generate three DNDP instances with a budget of 25%, 50% and 75% of the total cost of opening all links. This constitutes a dataset of a total of 180 DNDP instances including 60 instances of each transportation network.

All algorithms are implemented in Python on a Windows machine with a i9 CPU at 3.19 GHz and 64 GB of memory. All LPs and MILPs are solved using CPLEX 22.1 MIP solver (International Business Machines Corporation 2024) with a single thread. We set an optimality gap of 1% for algorithm convergence and we set a runtime limit of 1 hour for each instance. For reproducibility, all data and codes are made available at `https://github.com/davidrey123/DNDP-path`.

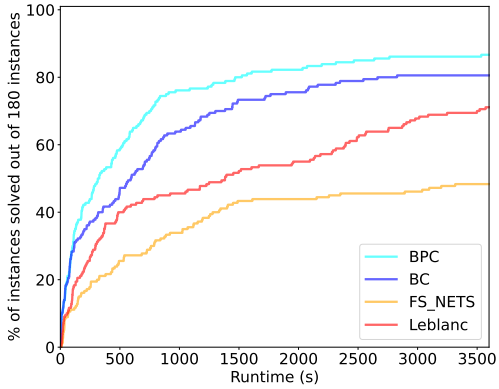
We report the results of the computational benchmarking of DNDP algorithms in Section 6.2. A detailed analysis on the behavior of the BPC algorithm is reported in Section 6.3.

6.2 Computational Benchmarking

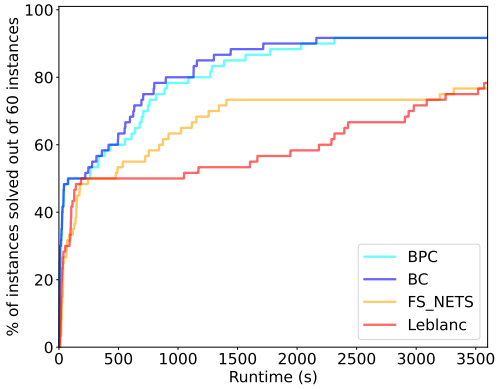
The main results of this study are reported in Figure 1 which depicts performance profile-like curves (Dolan and Moré 2002) of the four DNDP algorithms implemented, i.e. BPC, BC, FS_NETS and Leblanc. Specifically, for each algorithm, we report the percentage of instances solved over runtime. Figure 1a report performance profiles over all 180 instances considered. Performance profiles over the 60 instances of each network are reported in Figures 1b, 1c and 1d for SF, EM and BMC

networks, respectively. This benchmarking reveals that, overall, BPC dominates all other three DNDP algorithms. It is able to solve to optimality over 50% of the instances within 315s while BC and Leblanc require 591s and almost 1383s to solve the same percentage of instances, respectively. Within the 1 hour runtime limit, BPC is able to solve 86.7% of the 180 instances whereas BC and Leblanc achieve 80.3% and 71.1%, respectively. In contrast, our implementation of FS_NETS is able to solve only 48.3% of the 180 instances considered within the runtime limit. A closer look at network-based performance sheds several insights. On SF instances, BC tends to slightly dominate BPC. We also find that FS_NETS dominates Leblanc for a significant range of runtimes. This highlights that, for small networks, the link-based multicommodity network flow model which is used in both BC and FS_NETS provides a viable alternative to path-based counterparts. EM instances reveal a different pattern: here FS_NETS is dominated by all other algorithms while BPC slightly dominates BC. This emphasizes the gains obtained by exploiting the separability of the leader objective function to generate OA cuts within the BC and BPC algorithms. Results on BMC instances—which is the largest network tested—demonstrate the substantial benefits of the BPC algorithm over the benchmarks considered. BPC is found to require 582s for solving 50% of these instances—which corresponds to 10-link BMC instances—and is able to solve 68.3% of BMC instances within the runtime limit. Leblanc is the second-best performing algorithm and requires 2096s to solve 50% of these instances. BC ranks third and requires 2545s to solve 50% of these instances; while FS_NETS only manages to solve 11.7% of these instances within the 1 hour runtime limit.

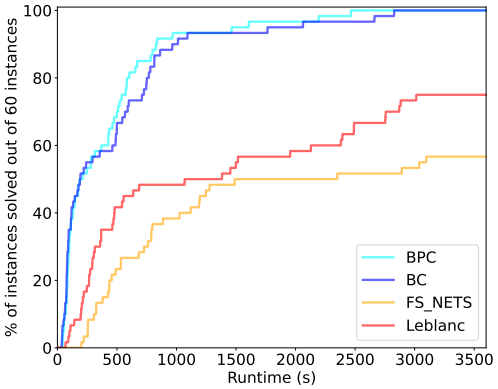
For a more detailed outlook, we organize instances in classes. Each class corresponds to a network (SF, EM or BMC), a number of $\mathcal{A}_2$ link variables (10 or 20) links and a budget level (25%, 50% or 75%). We focus on the first three instances (identified by ID) of each class and we report the UB and optimality gap (Gap, in percentage) upon termination, the runtime in seconds (Time) and the number of TAPs solved during execution (nTAP); for each of the four DNDP algorithms. Table 2 summarizes SF instances results, Table 3 summarizes EM instances results and Table 4 summarizes BMC instances results. Best performance is highlighted in boldface and rows in italics indicate that the 1 hour runtime limit was attained. Examining these tables reveal that BPC and BC strictly dominate FS_NETS and Leblanc for SF and EM instances. BPC and BC are able to solve all 20 SF instances. In turn, FS_NETS and Leblanc occasionally fail to solve SF 20-link instances within the 1 hour runtime limit but achieve optimality gaps lower than 5%. Behavior on EM 20-link instances favor BPC over BC in terms of runtime performance. We note that FS_NETS frequently times out on EM 20-link instances, while Leblanc tends to performs better. For SF and EM 10-link instances, few TAPs are solved and this contributes to competitive runtime overall. SF 20-link instances requires a significantly larger number of TAPs, i.e. several thousands of solves when using BPC, BC or Leblanc. FS_NETS requires substantially fewer TAP solves which can be attributed to its nested search tree implementation. Overall, EM instances requires significantly fewer TAP solves, i.e. by one or two orders of magnitude, than SF instances. For BMC 10-link instances, substantial runtime savings of about one order of magnitude are obtained by BPC compared to BC. Leblanc strikes a balance between BPC and BC on BMC instances although it is occasionally significantly slower than BPC on certain BMC 20-link instances. Further, on these instances, if timing out, both BPC and Leblanc achieve less-than-5% optimality gaps. In comparison, BC and FS_NETS systematically time out on BMC 20-link instances and return relatively large optimality gaps of the order of 30%-40%. This highlights the effectiveness of path-based approaches over link-based counterparts for larger scale DNDP instances.



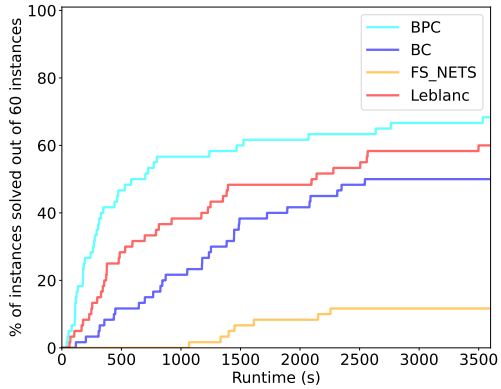
(a) All instances



(b) SF instances



(c) EM instances



(d) BMC instances

Figure 1: Performance profiles of DNDP algorithms: BPC, BC, FS_NETS and Leblanc.

6.3 BPC Algorithm Analysis

We investigate the impact of tuning algorithmic parameters within the BPC algorithm. For this analysis, we consider five BPC algorithm configurations whose names and features are listed in Table 5. We report the impact of three features that were found to affect algorithm behavior the most: the choice of the initialization heuristic among the considered (see Section 5.6), the threshold ϵ_{OA} used to generate OA cuts (see Section 5.3) and the use of interdiction and value function (VF) cuts (see Sections 5.4 and 5.5). The best-performing configuration is named `ls5`: this configuration uses the `LocalSearchKP` heuristic, a 5% OA cut threshold and no interdiction or VF cuts. We explore alternate configurations: `kb5` uses the `kBestKP` initialization heuristic; `ls10` and `ls1` use OA cuts thresholds of 10% and 1%, respectively; and `ls5cuts` adds interdiction and VF cuts during search.

The results of this analysis are summarized in Table 6: we focus on six DNDP instances, two per network including one with 10 $\mathcal{A}_2$ links and one with 20 $\mathcal{A}_2$ links. For each of these six instances, we

			Instance	BPC				BC				FS_NETS				Leblanc			
Network	$\|\mathcal{A}_2\|$	$B_{\%}$	ID	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP
SF	10	25	1	6219.7	0.63	**3.3**	11	6219.7	0.90	3.4	13	6219.7	0.00	20.1	7	6219.7	0.01	18.1	14
SF	10	25	2	6530.4	0.88	5.0	19	6530.4	0.88	**4.3**	19	6530.4	0.75	24.8	13	6530.4	0.81	13.8	13
SF	10	25	3	6219.5	0.94	10.3	29	6219.5	0.97	**9.8**	30	6219.5	0.29	32.9	15	6219.5	0.80	25.9	28
SF	10	50	1	5685.9	0.99	**18.8**	43	5685.9	0.96	20.3	46	5685.9	0.95	68.9	12	5685.9	0.81	99.7	52
SF	10	50	2	5763.1	0.94	3.9	9	5763.1	0.94	**3.4**	9	5763.1	1.00	34.7	8	5763.1	0.77	14.9	8
SF	10	50	3	5447.8	0.86	**24.5**	91	5447.8	0.97	29.1	100	5447.8	0.98	149.2	59	5447.8	0.99	103.3	76
SF	10	75	1	5283.1	0.97	6.3	16	5283.1	0.97	**6.1**	16	5283.1	0.50	84.5	16	5283.1	0.97	89.1	34
SF	10	75	2	5084.4	0.00	2.2	5	5084.4	0.00	**2.0**	5	5084.4	0.00	22.0	6	5084.4	0.00	9.0	5
SF	10	75	3	5071.4	1.00	**37.4**	133	5071.4	0.95	37.8	137	5071.4	0.99	176.6	125	5071.4	0.94	129.0	109
SF	20	25	1	5181.3	1.00	555.8	1484	5181.3	1.00	**498.0**	1465	5181.3	0.99	1343.4	435	5181.3	1.00	3520.5	1663
SF	20	25	2	5049.4	0.99	610.3	1639	5049.4	0.99	**495.4**	1613	5049.4	1.00	873.3	140	5049.4	1.00	2396.6	1722
SF	20	25	3	5235.4	0.99	747.8	2033	5235.4	0.99	**624.1**	2053	5251.7	0.99	1257.9	484	5235.4	1.00	2936.8	2053
SF	20	50	1	4286.6	1.00	1835.1	4300	4286.6	1.00	**1715.1**	4213	*4286.7*	*2.89*	*3601.0*	*310*	*4286.6*	*4.93*	*3601.6*	*1205*
SF	20	50	2	4117.2	1.00	384.1	973	4117.2	1.00	**360.8**	1081	4117.2	0.93	477.7	90	4117.2	1.00	3088.6	1675
SF	20	50	3	4307.9	1.00	1774.9	4090	4307.9	1.00	**1442.6**	4045	*4345.5*	*2.36*	*3601.1*	*587*	*4307.9*	*3.70*	*3600.0*	*1746*
SF	20	75	1	3904.0	1.00	816.9	2194	3904.0	1.00	**692.6**	2168	*3904.0*	*1.03*	*3602.2*	*1464*	3904.0	0.99	2313.7	1296
SF	20	75	2	3918.3	1.00	679.0	2801	3918.3	1.00	**557.9**	2736	3918.3	1.00	3198.5	2476	3918.3	1.00	1666.8	2500
SF	20	75	3	4038.7	1.00	1567.9	5381	4038.7	1.00	**1302.0**	5384	*4038.7*	*2.82*	*3600.1*	*1507*	*4038.7*	*1.22*	*3600.1*	*4396*

Table 2: Benchmarking of BPC, BC, FS_NETS and Leblanc's algorithms on SF network instances: performance is reported on the three first instances of each instance class. Optimality gaps (Gap) are reported in percentage and runtimes (Time) in seconds. Rows in italics indicate that the runtime limit was attained. Best performance among algorithms that converged before the runtime limit is highlighted in boldface.

report the performance of the five BPC algorithm configurations listed in Table 5. From left to right, the header of Table 6 reports: the UB and the optimality gap upon convergence, the root node LB (LB(0)). The next five columns summarize runtime data in seconds including, total, RMP, pricing (Prc.), OA cut generation (OA) and TAP solves (TAP) runtimes. The two right-most columns report the number of BPC iterations (It.) and the number of TAPs solved (nTAP). For each instance, we highlight best performance, i.e. largest root node LB, lowest total runtime, lowest number of iterations in boldface. Rows in italics corresponds to runs that attained the runtime limit of 1 hour and this data is excluded from consideration in terms of best performance.

The best performance overall in terms of runtime is obtained by configuration `ls5`: it outperforms other configurations for all but the 10-link EM instance. In turn, we find that `ls1` and `ls5cuts` fail to solve the 20-link SF instance. The majority of the runtime is spent solving TAPs. For 20-link instances, the second most time-consuming procedure of the BPC algorithm is solving the RMP (8). On SF and EM instances, the pricing runtime is of the order of 1-10s for 10-link instances and of 100s for 20-link instances. The runtime of this procedure is larger by one order of magnitude larger for BMC instances which can be attributed to the larger size of this network. OA cut generation does not require much computational effort relative to the other procedures of the BPC algorithm. We observe that reducing the OA cut generation threshold from 5% to 1% yields tighter root node LBs as shown by the performance of `ls1`. This also tends to reduce the number of iterations and the number of TAPs solved. Adding interdiction and VF cuts also tends to reduce the number iterations. In particular, on BMC instances, `ls5cuts` achieves the lowest number of iterations among the five configurations tested.

To further explore the behavior of the CG component of the BPC algorithm, we graph the number of path variables over BPC iterations using configuration `ls5` for the six instances considered. The results are shown in Figure 2 where the x-axis represent the progression of the BPC algorithm as the percentage of BPC iterations until convergence. This figure reveals that, for all six instances tested, the number of paths tends to increase sub-linearly during the course of the algorithm. For all three

	Instance			BPC				BC				FS_NETS				Leblanc			
Network	$\|\mathcal{A}_2\|$	$B_{\%}$	ID	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP
EM	10	25	1	821.5	0.00	**81.9**	6	821.5	0.00	113.3	8	821.5	0.77	255.2	2	821.5	0.00	196.0	5
EM	10	25	2	766.1	0.00	43.3	2	766.1	0.00	**36.1**	2	766.1	0.00	256.2	3	766.1	0.00	109.2	2
EM	10	25	3	742.3	0.66	**79.1**	4	742.3	0.66	80.8	4	742.3	0.00	327.9	4	742.3	0.63	274.7	1
EM	10	50	1	567.9	0.77	72.9	4	567.9	0.77	**65.9**	4	567.9	0.00	683.2	4	567.9	0.22	213.9	2
EM	10	50	2	554.6	0.00	84.2	5	554.6	0.00	**77.9**	5	554.6	0.00	424.6	5	554.6	0.00	205.2	3
EM	10	50	3	568.3	0.46	115.6	7	568.3	0.46	**112.2**	7	568.3	0.00	760.5	5	568.3	0.81	367.7	6
EM	10	75	1	521.5	0.98	132.1	10	521.5	0.98	**118.3**	10	521.5	0.71	1278.9	4	521.5	0.91	359.7	6
EM	10	75	2	516.2	0.48	78.7	7	516.2	0.71	**63.8**	6	516.2	0.81	787.1	4	516.2	0.87	199.1	3
EM	10	75	3	514.9	0.98	91.4	7	514.9	0.99	**80.8**	7	514.9	0.00	727.2	5	514.9	0.64	306.2	2
EM	20	25	1	917.7	0.75	**2194.3**	143	917.7	0.76	2827.2	142	*957.8*	*37.75*	*3766.5*	*2*	*940.8*	*22.18*	*3607.8*	*57*
EM	20	25	2	708.9	0.86	**427.5**	22	708.9	0.87	488.5	21	*708.9*	*1.56*	*3653.6*	*2*	708.9	0.86	2489.2	14
EM	20	25	3	1163.4	0.74	**427.1**	24	1163.4	0.71	499.0	25	*1163.4*	*24.17*	*3630.3*	*2*	1163.4	0.86	2380.2	29
EM	20	50	1	615.7	1.00	**2127.1**	161	615.7	1.00	2662.3	140	*652.8*	*15.13*	*3631.5*	*2*	*669.7*	*16.86*	*3609.5*	*42*
EM	20	50	2	570.0	1.00	**604.1**	37	570.0	0.98	750.1	38	*571.6*	*1.10*	*3618.2*	*2*	*570.0*	*1.88*	*3615.0*	*23*
EM	20	50	3	699.5	0.99	**1607.5**	114	699.5	0.96	2062.4	110	*718.8*	*21.11*	*3634.5*	*2*	*700.8*	*10.92*	*3610.4*	*50*
EM	20	75	1	522.1	0.93	**368.1**	23	522.1	1.00	499.5	29	*523.4*	*1.60*	*3601.5*	*2*	522.1	0.94	2393.3	23
EM	20	75	2	518.8	0.96	314.1	21	518.8	0.96	**240.3**	16	520.4	0.69	3038.5	3	518.8	0.93	1954.7	13
EM	20	75	3	549.9	0.95	**655.3**	42	549.9	0.98	709.4	43	*554.1*	*2.16*	*3614.1*	*2*	549.9	2.07	3610.2	32

Table 3: Benchmarking of BPC, BC, FS_NETS and Leblanc's algorithms on EM network instances: performance is reported on the three first instances of each instance class. Optimality gaps (Gap) are reported in percentage and runtimes (Time) in seconds. Rows in italics indicate that the runtime limit was attained. Best performance among algorithms that converged before the runtime limit is highlighted in boldface.

networks, the 20-link instance requires more path variables than the 10-link instance. The largest number of path variables handled by the RMP is observed for the 20-link EM instance and is of the order of 12,000. This illustrates the potential of the path-based formulation of the OA relaxation of SO-DNDP and the proposed CG procedure to solve DNDP instances at scale.

7 Conclusion

In this study, we presented a novel single tree branch-and-price-and-cut (BPC) algorithm for discrete network design problems under traffic equilibrium or DNDPs for short. DNDPs are notoriously challenging optimization problems which admit a natural Stackelberg game formulation in the presence of traffic equilibrium constraints. In this bilevel optimization formulation the leader represents the network designer while the follower represents a parameterized traffic equilibrium problem. In transportation, Wardrop's user equilibrium is often selected to model network users' route choice under congestion effects. Exploiting the separability and the convexity of the leader objective function, we introduce a new outer approximation (OA) scheme for the system-optimum (SO)-DNDP which corresponds to the high-point relaxation of the DNDP. Combining these successive relaxations of the bilevel problem with the path-based formulation of the DNDP, leads to a linear programming formulation that can be solved efficiently by column generation (CG). We develop a BPC framework to implement our approach and propose initialization techniques, cut generation rules and interdiction and value function cuts for algorithmic tuning.

We validate the performance of our BPC algorithm through comprehensive computational experiments over 180 problem instances based on three transportation networks of varying sizes. We use three alternative methods to compare the performance of the BPC algorithm including its branch-and-cut (BC) counterpart where a link-based multicommodity network model is used instead of the path-based network model. We show that our BPC algorithm outperform BC and existing

		Instance		BPC				BC				FS_NETS				Leblanc			
Network	$\|\mathcal{A}_2\|$	$B_{\%}$	ID	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP	UB	Gap	Time	nTAP
BMC	10	25	1	2620.3	0.68	**313.3**	92	2620.3	0.00	1234.5	372	*2628.7*	*36.93*	*3606.1*	*2*	2620.3	0.87	531.8	88
BMC	10	25	2	3088.1	0.00	**40.8**	2	3088.1	0.00	117.7	41	3088.1	0.00	1068.1	3	3088.1	0.00	63.2	2
BMC	10	25	3	2605.8	0.79	**192.8**	52	2605.8	0.00	871.3	321	*2605.8*	*37.67*	*3610.0*	*2*	2605.8	1.00	477.1	54
BMC	10	50	1	2573.0	0.95	**410.9**	156	2573.0	0.00	1722.1	512	*2573.0*	*35.83*	*3613.4*	*2*	2573.0	0.93	1353.0	154
BMC	10	50	2	2757.5	0.81	**123.3**	28	2757.5	0.69	696.4	263	*2763.0*	*41.81*	*3612.5*	*2*	2757.5	0.61	377.9	29
BMC	10	50	3	2570.8	0.98	**292.4**	90	2570.8	0.00	1485.6	512	*2570.8*	*39.62*	*3683.5*	*2*	2570.8	0.84	814.3	106
BMC	10	75	1	2573.0	0.95	**442.6**	156	2573.0	0.00	2087.3	652	*2573.0*	*35.21*	*3608.8*	*2*	2573.0	0.93	1383.1	154
BMC	10	75	2	2588.0	0.93	**112.0**	31	2588.0	0.00	1176.4	490	*2592.4*	*38.60*	*3621.1*	*2*	2588.0	0.82	343.4	30
BMC	10	75	3	2570.8	0.98	**304.3**	101	2570.8	0.00	1892.2	703	*2570.8*	*40.25*	*3608.2*	*2*	2570.8	0.84	1226.1	112
BMC	20	25	1	2730.8	1.00	**582.0**	83	*2730.8*	*31.02*	*3600.6*	*1404*	*2730.8*	*40.86*	*3605.8*	*2*	2730.8	0.98	2096.4	72
BMC	20	25	2	2698.2	0.99	**723.4**	184	*2698.2*	*26.94*	*3600.8*	*1614*	*2774.4*	*40.01*	*3603.8*	*2*	2698.2	0.97	2139.0	173
BMC	20	25	3	*2603.5*	*1.68*	*3601.3*	*1217*	*2603.5*	*28.72*	*3601.6*	*1612*	*2620.0*	*40.93*	*3604.6*	*2*	*2603.5*	*3.27*	*3604.4*	*212*
BMC	20	50	1	2581.4	0.99	**1822.3**	675	*2581.4*	*28.91*	*3602.0*	*1605*	*2581.4*	*37.41*	*3605.7*	*2*	*2581.4*	*2.02*	*3600.3*	*319*
BMC	20	50	2	2574.0	0.99	**1524.8**	524	*2574.0*	*27.68*	*3600.8*	*1563*	*2574.0*	*36.06*	*3611.2*	*2*	2574.0	1.00	3498.3	494
BMC	20	50	3	*2575.8*	*3.01*	*3600.4*	*1359*	*2575.8*	*29.16*	*3601.6*	*1540*	*2581.0*	*40.33*	*3610.5*	*2*	*2575.8*	*4.06*	*3611.8*	*271*
BMC	20	75	1	2573.7	1.00	**2763.7**	925	*2573.7*	*28.94*	*3600.5*	*1567*	*2573.7*	*37.91*	*3610.1*	*2*	*2573.7*	*2.15*	*3604.8*	*374*
BMC	20	75	2	*2571.3*	*1.07*	*3600.0*	*1275*	*2571.3*	*28.69*	*3600.7*	*1489*	*2571.3*	*36.31*	*3607.4*	*2*	*2571.3*	*2.79*	*3604.2*	*404*
BMC	20	75	3	*2571.5*	*3.55*	*3600.2*	*1265*	*2571.5*	*29.36*	*3602.1*	*1529*	*2571.8*	*40.33*	*3605.1*	*2*	*2571.5*	*4.26*	*3603.9*	*280*

Table 4: Benchmarking of BPC, BC, FS_NETS and Leblanc's algorithms on BMC network instances: performance is reported on the three first instances of each instance class. Optimality gaps (Gap) are reported in percentage and runtimes (Time) in seconds. Rows in italics indicate that the runtime limit was attained. Best performance among algorithms that converged before the runtime limit is highlighted in boldface.

Config.	Init. heuristic	OA cuts threshold	Interdiction and VF cuts
`ls5`	`LocalSearchKP`	5%	No
`kb5`	`kBestKP`	5%	No
`ls10`	`LocalSearchKP`	10%	No
`ls1`	`LocalSearchKP`	1%	No
`ls5cuts`	`LocalSearchKP`	5%	Yes

Table 5: Names and features of BPC algorithm configurations tested.

approaches in the literature. Notably, we demonstrate that our BPC algorithm is efficient on both small and larger scale problem instances whereas other DNDP algorithms either fail to scale-up due to their reliance on link-based multicommodity network models (i.e. BC and FS_NETS) or to the inherent structure (Leblanc). In contrast, the BPC algorithm is able to consistently solve—or achieve competitive optimality gaps on—problem instances of varying number of variable links and/or network features. For reproducibility purposes, all data and codes used in this study are made publicly available at `https://github.com/davidrey123/DNDP-path`. For presentation and experimentation purposes, we focused on the link addition DNDP which is the most studied DNDP in the literature. We emphasize that most of the methods developed can be immediately applied to other DNDPs such as mixed discrete-continuous DNDPs or node-addition DNDPs.

This research can be extended in several directions. From a methodological standpoint, further research may explore the integration of additional cuts or penalty methods to reduce the optimality gap during search. While the value function cuts considered tend to reduce the number of BPC iterations, their incorporation leads to excessive computational efforts. Techniques to mitigate these effects could be explored. From a modeling perspective, this study focused on discrete NDPs, however the proposed OA relaxations and the CG approach can be applied to continuous NDPs as well. In

Instance								Time (s)						
Network	$\|\mathcal{A}_2\|$	$B_{\%}$	ID	Config.	UB	Gap (%)	LB(0)	Total	RMP	Prc.	OA	TAP	It.	nTAP
SF	10	50	1	`ls5`	5685.9	0.99	4829.2	**18.8**	4.3	1.2	0.8	13.0	197	43
				`kb5`	5685.9	1.00	4826.1	20.1	4.4	1.2	1.6	12.8	199	44
				`ls10`	5685.9	0.98	4810.3	23.6	3.4	1.4	1.0	17.7	187	51
				`ls1`	5685.9	0.89	**4836.3**	26.7	11.8	1.4	1.0	12.3	**169**	39
				`ls5cuts`	5685.9	0.99	4829.2	20.9	5.8	1.2	0.8	12.8	177	43
SF	20	50	1	`ls5`	4286.6	1.00	**3559.3**	**1835.1**	710.0	73.7	3.4	950.2	19057	4300
				`kb5`	4286.6	1.00	3555.1	1916.6	719.6	76.4	5.4	1005.7	**18841**	4262
				`ls10`	4286.6	1.00	3554.8	2039.7	621.2	89.6	3.0	1213.7	19641	4592
				`ls1`	*4286.6*	*1.87*	*3571.7*	*3600.0*	*2645.8*	*68.8*	*7.1*	*791.3*	*13857*	*2843*
				`ls5cuts`	*4286.7*	*1.03*	*3559.3*	*3600.0*	*2861.3*	*48.7*	*3.1*	*603.5*	*12058*	*2593*
EM	10	50	1	`ls5`	567.9	0.77	487.1	72.9	3.2	9.0	7.0	45.0	**23**	4
				`kb5`	567.9	0.17	487.4	70.7	4.0	10.9	9.7	45.8	29	4
				`ls10`	567.9	0.85	485.9	77.3	3.3	11.8	8.2	53.7	25	4
				`ls1`	567.9	0.71	**487.6**	**64.2**	3.4	8.8	6.9	44.9	**23**	4
				`ls5cuts`	567.9	0.81	487.1	81.0	4.5	9.9	7.0	45.6	**23**	4
EM	20	50	1	`ls5`	615.7	1.00	489.9	**2127.1**	332.2	76.6	12.4	1702.0	1477	161
				`kb5`	615.7	1.00	487.6	2331.3	423.5	88.7	15.5	1799.2	1479	156
				`ls10`	615.7	1.00	488.7	2328.9	333.6	93.8	13.8	1884.1	**1411**	149
				`ls1`	615.7	0.99	**490.2**	2780.4	945.6	88.7	13.5	1723.1	1503	146
				`ls5cuts`	615.7	0.99	489.9	2643.6	714.4	79.1	13.2	1817.0	1459	170
BMC	10	50	1	`ls5`	2573.0	0.95	2420.1	**410.9**	16.2	42.9	12.5	338.1	423	156
				`kb5`	2573.0	0.99	2418.6	420.1	16.5	44.6	14.0	344.0	421	156
				`ls10`	2573.0	0.93	2416.7	504.0	17.6	52.7	15.2	417.3	441	164
				`ls1`	2573.0	0.91	**2422.6**	502.3	38.5	49.7	13.7	398.7	415	152
				`ls5cuts`	2573.0	0.95	2420.1	453.2	39.5	45.4	13.1	351.1	**381**	156
BMC	20	50	1	`ls5`	2581.4	0.99	2417.5	**1822.3**	174.8	242.5	23.2	1375.6	2225	675
				`kb5`	2581.4	1.00	2413.6	2094.5	263.7	310.0	23.9	1488.7	2783	728
				`ls10`	2581.4	0.99	2414.9	2702.9	215.4	359.5	26.9	2092.8	2743	827
				`ls1`	2581.4	1.00	**2422.5**	2198.3	436.8	274.2	29.5	1448.0	2245	653
				`ls5cuts`	2581.4	1.00	2417.5	2403.4	653.7	254.2	24.3	1460.0	**2131**	662

Table 6: Analysis of the BPC algorithm under multiple parameter configurations. Six DNDP instances are selected: two per network including one with $|\mathcal{A}_2| = 10$ and one with $|\mathcal{A}_2| = 20$. Rows in italics indicate that the runtime limit was attained. Best performance among algorithms that converged before the runtime limit is highlighted in boldface.

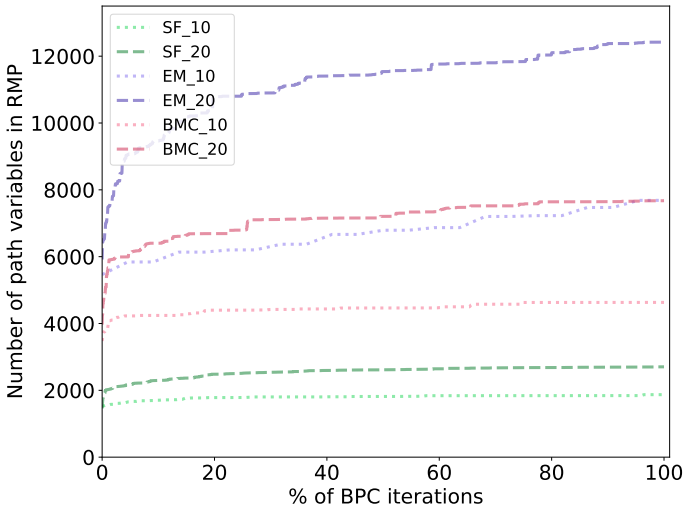


Figure 2: Analysis of the column generation procedure of the BPC algorithm for the best performing configuration. Six DNDP instances are selected: two per network including one with $|\mathcal{A}_2| = 10$ and one with $|\mathcal{A}_2| = 20$.

this context, the branch-and-bound framework may be omitted and continuous NDPs could benefit from exploiting the OA and CG procedures developed in this study. DNDPs under traffic equilibrium have several practical applications notably in transportation networks but also in telecommunications networks (Correa and Stier-Moses 2011). Discrete NDPs arising in these contexts can benefit from the proposed approach by adapting its core elements to specific problem contexts and also extend to other discrete problems such as bilevel facility location or network operation scheduling problems under Wardropian equilibria.

References

Abdulaal M, LeBlanc LJ (1979) Continuous equilibrium network design models. *Transportation Research Part B: Methodological* 13(1):19–32.

Bagloee SA, Sarvi M, Patriksson M (2017) A hybrid branch-and-bound and benders decomposition algorithm for the network design problem. *Computer-Aided Civil and Infrastructure Engineering* 32(4):319–343.

Bar-Gera H (2010) Traffic assignment by paired alternative segments. *Transportation Research Part B: Methodological* 44(8-9):1022–1046.

Beckmann M, McGuire CB, Winsten CB (1956) Studies in the economics of transportation. Technical report.

Braess D (1968) Über ein paradoxon aus der verkehrsplanung. *Unternehmensforschung* 12(1):258–268.

Colson B, Marcotte P, Savard G (2007) An overview of bilevel optimization. *Annals of Operations Research* 153(1):235–256.

Correa JR, Schulz AS, Stier-Moses NE (2004) Selfish routing in capacitated networks. *Mathematics of Operations Research* 29(4):961–976.

Correa JR, Stier-Moses NE (2011) Wardrop equilibria. *Encyclopedia of Operations Research and Management Science. Wiley* .

Dafermos S (1980) Traffic equilibrium and variational inequalities. *Transportation Science* 14(1):42–54.

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. *Operations Research* 8(1):101–111.

Dial RB (1971) A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation Research* 5(2):83–111.

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. *Mathematical programming* 91:201–213.

Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming* 36(3):307–339.

Farvaresh H, Sepehri MM (2011) A single-level mixed integer linear formulation for a bi-level discrete network design problem. *Transportation Research Part E: Logistics and Transportation Review* 47(5):623–640.

Farvaresh H, Sepehri MM (2013) A branch and bound algorithm for bi-level discrete network design problem. *Networks and Spatial Economics* 13:67–106.

Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming* 66:327–349.

Fontaine P, Minner S (2014) Benders decomposition for discrete–continuous linear bilevel problems with application to traffic network design. *Transportation Research Part B: Methodological* 70:163–172.

Gairing M, Harks T, Klimm M (2017) Complexity and approximation of the continuous network design problem. *SIAM Journal on Optimization* 27(3):1554–1582.

Gao Z, Wu J, Sun H (2005) Solution algorithm for the bi-level discrete network design problem. *Transportation Research Part B: Methodological* 39(6):479–495.

International Business Machines Corporation (2024) IBM ILOG CPLEX Optimization Studio. URL https://www.ibm.com/products/ilog-cplex-optimization-studio.

Leblanc LJ (1975) An algorithm for the discrete network design problem. *Transportation Science* 9(3):183–199.

Lozano L, Smith JC (2017) A value-function-based exact approach for the bilevel mixed-integer programming problem. *Operations Research* 65(3):768–786.

Luathep P, Sumalee A, Lam WH, Li ZC, Lo HK (2011) Global optimization method for mixed transportation network design problem: a mixed-integer linear programming approach. *Transportation Research Part B: Methodological* 45(5):808–827.

Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Operations Research* 53(6):1007–1023.

Magnanti TL, Wong RT (1984) Network design and transportation planning: Models and algorithms. *Transportation Science* 18(1):1–55.

Mahmassani HS, Chang GL (1987) On boundedly rational user equilibrium in transportation systems. *Transportation Science* 21(2):89–99.

Rey D (2020) Computational benchmarking of exact methods for the bilevel discrete network design problem. *Transportation Research Procedia* 47:11–18.

Roughgarden T (2006) On the severity of Braess's paradox: Designing networks for selfish users is hard. *Journal of Computer and System Sciences* 72(5):922–953.

Roughgarden T, Tardos É (2002) How bad is selfish routing? *Journal of the ACM (JACM)* 49(2):236–259.

Sheffi Y (1985) *Urban transportation networks*, volume 6 (Prentice-Hall, Englewood Cliffs, NJ).

Transportation Networks for Research Core Team (2024) Transportation Networks for Research. URL https://github.com/bstabler/TransportationNetworks.

Wang DZ, Liu H, Szeto W (2015) A novel discrete network design problem formulation and its global optimization solution algorithm. *Transportation Research Part E: Logistics and Transportation Review* 79:213–230.

Wang S, Meng Q, Yang H (2013) Global optimization methods for the discrete network design problem. *Transportation Research Part B: Methodological* 50:42–60.

Wardrop JG (1952) Some theoretical aspects of road traffic research. *Inst Civil Engineers Proc London/UK/*.

Xie J, Xie C (2016) New insights and improvements of using paired alternative segments for traffic assignment. *Transportation Research Part B: Methodological* 93:406–424.

Yang H, H Bell MG (1998) Models and algorithms for road network design: a review and some new developments. *Transport Reviews* 18(3):257–278.