Column Elimination for Capacitated Vehicle Routing Problems

Anthony Karahalios$^{[0000-0001-9479-4080]}$ and Willem-Jan van Hoeve$^{[0000-0002-0023-753X]}$

Carnegie Mellon University, Pittsburgh PA 15213, USA
{akarahal,vanhoeve}@andrew.cmu.edu

Abstract. We introduce a column elimination procedure for the capacitated vehicle routing problem. Our procedure maintains a decision diagram to represent a relaxation of the set of feasible routes, over which we define a constrained network flow. The optimal solution corresponds to a collection of paths in the decision diagram and yields a dual bound. The column elimination process iteratively removes infeasible paths from the diagram to strengthen the relaxation. The network flow model can be solved as a linear program with a conventional solver or via a Lagrangian relaxation. To solve the Lagrangian subproblem more efficiently, we implement a special successive shortest paths algorithm. We introduce several cutting planes to strengthen the dual bound, including a new type of clique cut that exploits the structure of the decision diagram. We experimentally compare the bounds from column elimination with those from column generation for capacitated vehicle routing problems.

1 Introduction

The capacitated vehicle routing problem (CVRP) can be stated as follows [29]. Given a set of locations each with a specified weight and a fleet of trucks each with a specified capacity, the problem asks to design a route for each truck such that each location is visited by a truck, for each truck the total weight of its visited locations does not exceed the capacity, and the sum of the truck route lengths is minimized. It is a central problem in logistics and has become increasingly important over the last decade due to the rise of last-mile delivery applications. The CVRP is among the most studied NP-hard combinatorial optimization problems and finding provably optimal solutions remains a challenge in practice. Current state-of-the-art exact methods can solve up to around 200 locations optimally within a reasonable of time, with branch-cut-and-price (BCP) methods performing particularly well [12,5,23,24,25].

BCP is an effective method for solving generic large-scale integer programming models [7]. It relies on column generation to solve the linear programming relaxation: working with a restricted set of variables (or columns), column generation iteratively adds new variables to the model until an optimal basis is found. Despite its successes, column generation has some weaknesses. For example, it may take many iterations to converge to the optimal solution due to dual degeneracy of the intermediate solutions. Furthermore, branching decisions or cutting planes that strengthen the relaxation may complicate the pricing problem that finds new variables.

We study an alternative approach that does not rely on a pricing problem, thereby avoiding the potential drawbacks of column generation mentioned above. Instead of using a restricted set of columns, column elimination works with a *relaxed* set of columns, from which infeasible ones are iteratively eliminated. As the total number of columns can be exponentially large, we use relaxed decision diagrams to compactly represent and manipulate the set of columns. This method was first introduced for the graph coloring problem in [31,33,15], then applied to the traveling salesperson problem with a drone [28,27], and later termed 'column elimination' [32].

The main focus of this work is to develop strong *dual bounds* for the CVRP using column elimination. As will be formalized later, column elimination and column generation will produce the same dual bound if they work from the same underlying route relaxation. Column elimination can potentially produce stronger bounds than the initial route relaxation as it can remove infeasible columns beyond those that are excluded by the initial route relaxation (cf. [26]). Moreover, column elimination allows a more liberal use of cutting planes to strengthen the relaxation. We show how existing cuts from the column generation literature can be expressed directly into the column elimination model, while in addition the decision diagram representation of the columns permits us to develop new cuts. The novel contributions include introducing cuts to column elimination, developing an efficient solution method via a Lagrangian reformulation, and showing how column elimination can produce bounds competitive with state-of-the-art solvers for the CVRP.

The paper is organized as follows. In Section 2 we present the column formulation of the CVRP. Section 3 describes the decision diagram-based constrained network flow formulation. The column elimination procedure is presented in Section 4. Section 5 present our Lagrangian relaxation. In Section 6 we describe how cutting planes can be added to strengthen the model. Section 7 presents a reduced cost-based arc fixing procedure to reduce the model size. We conduct experimental results in Section 8 and conclude in Section 9.

2 Column Formulation for CVRP

We first give a formal definition of the CVRP [29]. Let $G = (V, A)$ be a complete directed graph with vertex set $V = \{0, 1, \dots, n\}$ and arc set $A = \{(i, j) \mid i, j \in V, i \neq j\}$. Vertex 0 represents the depot and vertices $\{1, \dots, n\}$ represent the locations to be visited. We will interchangeably use vertices and locations. Each vertex $i \in V$ has a demand $q_i \geq 0$ and each arc $a \in A$ has a length $l_a \geq 0$. Let K be the number of (homogeneous) vehicles, each with capacity Q. A *route* is a sequence of vertices $[v_1, v_2, \dots, v_k]$ starting and ending at the depot with total demand at most Q. The *distance* of a route is the sum of its arc lengths, i.e.,

$\sum_{i=1}^{k-1} l_{(v_i,v_{i+1})}$. The CVRP consists in finding K routes such that each vertex except for the depot belongs to exactly one route and the sum of the route distances is minimized.

The column formulation for the CVRP is based on the set R of all feasible elementary routes [6]. We let d_r denote the distance of route $r \in R$. We define an $n \times |R|$ matrix M such that $M_{ir} = 1$ if vertex $i \in \{1, 2, \ldots, n\}$ belongs to route $r \in R$, and $M_{ir} = 0$ otherwise. That is, each column vector in M corresponds to a route. Lastly, we define a binary decision variable x_r for each $r \in R$. The column formulation of the CVRP is:

$$\begin{aligned} \min & \sum_{r \in R} d_r x_r \\ \text{s.t.} & \sum_{r \in R} M_{ir} x_r = 1 \quad \forall i \in \{1, 2, \ldots, n\} \\ & \sum_{r \in R} x_r = K \\ & x_r \in \{0, 1\} \qquad \forall r \in R. \end{aligned} \tag{1}$$

This model is also known as the *set partitioning* formulation. In practice the set of routes R often has exponential size, which restricts the direct application of the set partitioning model to very small instances. Branch-and-price [7] provides a more scalable approach by using a column generation procedure to solve the continuous linear programming relaxation of (1).

Column generation starts by solving the linear programming relaxation of the set partitioning model defined on a (small) subset of variables, known as the *restricted master problem*. Using the dual variables of the optimal solution it then solves a *pricing problem* to find a new variable with a negative reduced cost. This process continues until no more improving variables exist and the restricted master has a provably optimal basis. To ensure integer feasibility, column generation is embedded into a systematic search.

Solving the pricing problem for the CVRP is not straightforward, because it corresponds to the NP-hard elementary shortest path problem with resource constraints [13]. It can be solved with dynamic programming, which is however limited by the exponential state space size. A computationally efficient alternative is to relax the pricing problem to find a shortest path that is not necessarily elementary, i.e., certain locations can be visited more than once [20]. Recent examples include the q-route relaxation [12] and the ng-route relaxation [5]. The linear programming model from route relaxations can be further strengthened by adding cutting planes to the restricted master problem [23].

3 Decision Diagram Formulation for CVRP

The key ingredient of the column elimination procedure is to compactly represent the set of routes R as a decision diagram. The CVRP can then be formulated as a constrained integer network flow over the decision diagram following the methodology in [33,27].

3.1 From Dynamic Programming to Decision Diagrams

For our purposes, a *decision diagram* is a layered acyclic weighted directed graph $D = (\mathcal{N}, \mathcal{A})$ with node set $\mathcal{N}$ and arc set $\mathcal{A}$. Each arc $a \in \mathcal{A}$ has an associated cost c_a and arc label ℓ_a. Graph D has a single root node r and a single terminal node t. While there are different methods to compile decision diagrams, we employ a generic approach that constructs a decision diagram from a dynamic programming formulation [8]. It requires a state definition, an (implied) set of states $\mathcal{S}$, a set of labels $\mathcal{L}$, a state transition function $f : (\mathcal{S} \times \mathcal{L}) \to \mathcal{S}$ and a transition cost function $g : (\mathcal{S} \times \mathcal{L}) \to \mathbb{R}$.

For the CVRP, we can use the dynamic programming formulation for the elementary shortest path problem with resource constraints [13], which we will refer to as DP_{ESPRC}. We define each state as a tuple (S, w, e) where $S \subseteq V$ represents the set of visited locations, $w \geq 0$ represents the accumulated 'weight', and $e \in V$ represents the last visited location. The initial state is defined as $(\varnothing, 0, 0)$. The set of labels is $\mathcal{L} = V$. Given a state $s = (S, w, e)$ and control (or label) $i \in V$ such that $i \notin S$ and $w + q_i \leq Q$, we define the transition function $f(s, i)$ as

$$f(s, i) = (S \cup \{i\}, w + q_i, i)$$

with associated transition cost function $g(s, i) = l_{(e,i)}$.

The decision diagram is defined similar to the state-transition graph of the dynamic programming model: the nodes in $\mathcal{N}$ correspond to the states and the arcs in $\mathcal{A}$ correspond to the transitions. That is, the root node r corresponds to the initial state $(\varnothing, 0, 0)$. For each transition $f(s_1, i) = s_2$ from state s_1 to state s_2 we define an arc $(u, v) \in \mathcal{A}$ where u corresponds to s_1 and v to s_2. The arc (u, v) has associated label $\ell_{(u,v)} = i$ and arc cost $c_{(u,v)} = g(s, i)$. We define the terminal node t as the collection of all states (S, w, e) with $|S| \geq 1$ and $e = 0$, i.e., t is the endpoint of all transitions that take label $i = 0$ to finish the route at the depot.

3.2 Dynamic Programming for Route Relaxations

The two most-used route relaxations for the CVRP in the column generation literature are the q-route relaxation [12] and the ng-route relaxation [5]. Both are based on the DP_{ESPRC} dynamic programming formulation, but relax the set of visited locations S.

The *q-route relaxation* maintains the last q visited locations. The dynamic program has state definition (SQ, w) where w is defined as above, and $SQ = [i_1, \ldots, i_q]$ is a sequence of locations. The initial state is $([\text{-}, \ldots, \text{-}], 0)$. Given a state $s = (SQ, w)$ and label $i \in V$ such that $i \notin SQ$ and $w + q_i \leq Q$, we define the transition function as

$$f^{\text{SQ}}(s, i) = ([i_2, \ldots, i_q, i], w + q_i)$$

with associated transition cost function $g^{\text{SQ}}(s, i) = l_{(i_q,i)}$. We denote the resulting dynamic programming model as DP_{SQ_q}.

For the *ng-route relaxation*, we assume that a set $N_i \subseteq V$ of size g exists for each $i \in \{1, \ldots, n\}$. The set N_i must include i and typically represents the g locations closest to i. As state definition, we use (NG, w, e) where the 'no-good' set $NG \subseteq V$ is a subset of visited locations, and w and e are as above. The initial state is $(\varnothing, 0, 0)$. Given a state $s = (NG, w, e)$ and label $i \in V$ such that $i \notin NG$ and $w + q_i \leq Q$, we define the transition function as

$$f^{\mathrm{NG}}(s, i) = ((NG \cup \{i\}) \cap N_i, w + q_i, i)$$

with associated transition cost function $g^{\mathrm{NG}}(s, i) = l_{(e,i)}$. We denote the resulting dynamic programming model as DP_{NG_g}. Observe that DP_{SQ_q} and DP_{NG_g} forbid cycles of length at most q and g, respectively.

3.3 Exact and Relaxed Decision Diagrams

We next specify the concepts of exact and relaxed decision diagrams [8] in the context of the CVRP. Given a decision diagram D, we let P_D denote the set of arc-label specified r-t paths in D. We slightly abuse notation and let c_p denote the sum of the arc costs of path $p \in P_D$. Recall that d_r represents the distance of route $r \in R$.

Definition 1. *Let R be a set of routes for the CVRP and let D be a decision diagram. We say that D is an* exact *diagram w.r.t. R if $P_D = R$ and $c_p = d_r$ for all $p \in P_D$, where r is the route representation of p. We say that D is a* relaxed *diagram w.r.t. R if $P_D \supseteq R$ and $c_p \leq d_r$ for all $p \in P_D$.*

Theorem 1. *The decision diagram derived from DP_{ESPRC} is exact w.r.t. R. The decision diagrams derived from DP_{SQ_q} and DP_{NG_g} are both relaxed w.r.t. R.*

Proof. Because DP_{ESPRC} encodes elementary paths and represents all possible feasible routes and their associated length, the resulting decision diagram is exact. Both DP_{SQ_q} and DP_{NG_g} encode a relaxation that contains elementary paths, and therefore represent a superset of all possible feasible routes. Because they both maintain the last visited location their cost functions are not relaxed, i.e., $c_p = d_r$ for each path p and associate route r (which is not necessarily elementary). Hence, DP_{SQ_q} and DP_{NG_g} yield a relaxed decision diagram. $\square$

3.4 Constrained Network Flow Formulation

We next reformulate the set partitioning model (1) as a constrained integer network flow model over a given decision diagram $D = (\mathcal{N}, \mathcal{A})$. We introduce a 'flow' variable $y_a \geq 0$ for each $a \in \mathcal{A}$. The set of incoming arcs of a node u is denoted by $\delta^-(u)$. Likewise $\delta^+(u)$ denotes the set of outgoing arcs of u. We denote the set of arcs in $\mathcal{A}$ with label i by $\mathcal{A}^i$. The model is as follows:

$$F(D): \quad \min \sum_{a \in \mathcal{A}} c_a y_a \tag{2}$$

$$\text{s.t.} \sum_{a \in \delta^-(u)} y_a - \sum_{a \in \delta^+(u)} y_a = 0 \qquad \forall u \in \mathcal{N} \setminus \{r, t\} \tag{3}$$

$$\sum_{a \in \mathcal{A}^i} y_a = 1 \qquad \forall i \in V \setminus \{0\} \tag{4}$$

$$\sum_{a \in \delta^+(r)} y_a = K \tag{5}$$

$$y_a \in \{0, 1\} \qquad \forall a \in \mathcal{A}. \tag{6}$$

The objective function (2) minimizes the sum of all arc costs. The 'flow conservation' constraints (3) ensure that the solution is a collection of labeled r-t paths. Constraints (4) ensure that all locations are visited once. Constraint (5) enforces that exactly K units of flow originate from r. The binary constraints (6) complete the formulation.

Theorem 2. *Let D be an exact decision diagram w.r.t. the set of routes R. Model $F(D)$ is an exact formulation of the CVRP.*

The proof relies on the fact that the dynamic programming model represents all possible routes, that each solution of the network flow model consists of exactly K r-t paths, and that each r-t path corresponds to a feasible route.

Corollary 1. *Let D be a relaxed decision diagram w.r.t. the set of routes R. Model $F(D)$ yields a dual bound for the CVRP.*

In the remainder of this paper, we will use the continuous linear programming relaxation of model $F(D)$, referred to as $LP(F(D))$, which is obtained by replacing the integrality constraints (6) by $0 \leq y_a \leq 1$ for all $a \in \mathcal{A}$.

4 Column Elimination Procedure

We present a schematic representation of column elimination in Figure 1. Starting with an initial relaxed decision diagram D, the column elimination procedure iteratively 1) solves the constrained network flow model $F(D)$, 2) decomposes the solution into paths (routes), 3) identifies infeasible paths and removes them from D, and repeats. The process terminates when no infeasible paths are detected in which case $F(D)$ is solved to optimality. It can also terminate earlier when the dual bound matches a given (or heuristically generated) primal bound, or when a different stopping criterion such as a time or memory limit is met. The procedure can utilize either the integer model $F(D)$ or its continuous relaxation $LP(F(D))$; using $LP(F(D))$ would solve the continuous linear programming relaxation of (1), but could be embedded in branch-and-bound to solve the full problem.

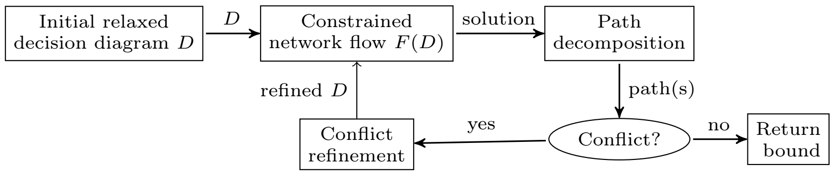


Fig. 1. Overview of the column elimination framework, adapted from [27].

Locations $V = \{0, 1, 2, 3, 4\}$
Depot $= 0$
Demands $q_1 = q_2 = q_3 = 1$, $q_4 = 2$
Number of trucks $K = 2$
Vehicle capacity $Q = 3$

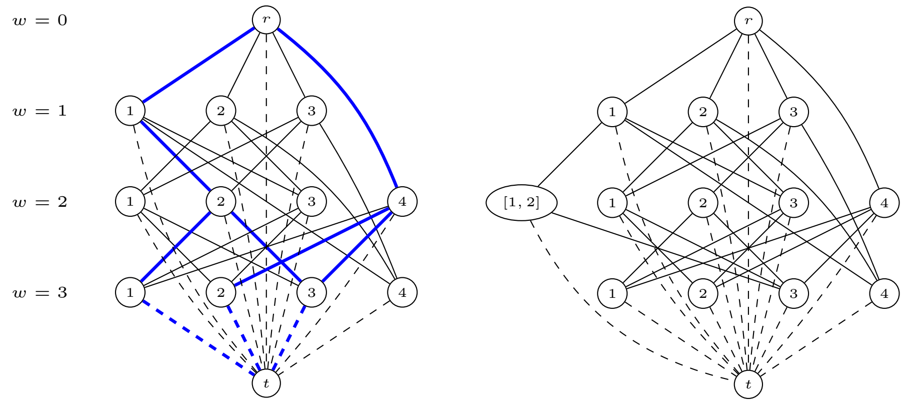
l_{ij}	0	1	2	3	4
0	0	5	10	5	10
1	5	0	10	10	15
2	10	10	0	10	15
3	5	10	10	0	10
4	10	15	15	10	0

Fig. 2. Input data for the CVRP instance in Example 1.

Any (existing) route relaxation for the CVRP can be applied to construct the initial relaxed decision diagram. Recall that model DP_{ESPRC} has state definition (S, w, e), and each of these three elements can potentially be relaxed to define a relaxed decision diagram. The q-route and ng-route relaxations only relax the elementarity constraint, i.e., the set S. This means that conflicts will only come in the form of repeated labels; each path respects the truck capacity constraint and the route costs are exact. For a decision diagram D derived from such a relaxation, $F(D)$ is an exact formulation for the CVRP. In practice, we prefer using a relaxation that is relatively small and provides a 'good' starting point in terms of bound quality from $LP(F(D))$. In our experiments, we therefore use DP_{SQ_q} with $q = 1$ and DP_{NG_g} with $g = 2$ to initialize the relaxed decision diagram, with DP_{NG_2} performing best.

Given the initial relaxed decision diagram D, we solve the associated model $LP(F(D))$, apply a path decomposition of the solution, and inspect the paths for any conflicts. For our choice of route relaxations, the only conflicts arise from repetition of locations along a path. To remove a conflict, we follow the (partial) path elimination process outlined in [33]: it essentially separates the path by introducing a new node at each layer, and removing the arc associated with the repeated label. During this process, we will update the state information of the nodes along the separated path. We illustrate conflict separation in the next example, and refer to [33] for more details.

Example 1. Consider the CVRP instance with the problem data given in Figure 2. The integer optimal solution uses routes $[0, 1, 2, 0]$ and $[0, 3, 4, 0]$ with total distance 50. The relaxed decision diagram based on DP_{SQ_q} with $q = 1$ is presented in Figure 3(a). Each node in the diagram is associated with its SQ state, i.e., the last visited location. The weights are omitted from the states; instead nodes with the same cumulative weight are represented in the same layer. For clarity, we also omit the arc labels and arc costs. Arcs into t correspond



a. Relaxed decision diagram from DP_{SQ_1}. b. Refined decision diagram.

Fig. 3. Decision diagrams for the CVRP instance in Example 1. Figure (a) depicts the relaxed decision diagram obtained from the q-route relaxation. The optimal solution to model $LP(F(D))$ is indicated by thick blue arcs. Figure (b) represents the refined decision diagram after eliminating the partial path $[1, 2, 1]$ that contains a conflict.

to terminating a route and are dashed. The optimal solution to the linear programming relaxation of $F(D)$ yields dual bound 48.333 and uses the following arc-label specified paths: path $(1, 2, 1, 0)$ with flow value $\frac{1}{3}$, path $(1, 2, 3, 0)$ with flow value $\frac{1}{3}$, path $(4, 2, 0)$ with flow value $\frac{1}{3}$, and path $(4, 3, 0)$ with flow value $\frac{2}{3}$.

The first path contains a conflict: label 1 is repeated. We separate this conflict by rerouting the path to a new node with state $SQ = [1, 2]$ memorizing location 1 in addition to 2. As a result, we eliminate the arc with label 1 from the new state. The refined decision diagram is depicted in Figure 3(b). It yields a dual bound of value 50, which is optimal.

5 Lagrangian Relaxation

Because the decision diagrams can grow large in size, solving the constrained network flow model can become the computational bottleneck of our method, even when we consider the continuous linear programming relaxation. To potentially solve the model more efficiently, we consider solving a Lagrangian relaxation, similar to [28,27], that has optimal bound equivalent to $LP(F(D))$. We obtain our Lagrangian relaxation of the constrained network flow model by dualizing constraints (4) that require that each location is visited once. We introduce a Lagrangian multiplier λ_i for each $i \in V$ ($\lambda_0 = 0$ is only introduced for notational ease), and define the Lagrangian relaxation as

$$L(D,\lambda): \quad \min \sum_{a\in\mathcal{A}} c_a y_a + \sum_{i\in V\setminus\{0\}} \lambda_i (1 - \sum_{a\in\mathcal{A}^i} y_a) \tag{7}$$

$$\text{s.t.} \sum_{a\in\delta^-(u)} y_a - \sum_{a\in\delta^+(u)} y_a = 0 \qquad \forall u \in \mathcal{N}\setminus\{r,t\} \tag{8}$$

$$\sum_{a\in\delta^+(r)} y_a = K \tag{9}$$

$$y_a \in \{0,1\} \qquad \forall a \in \mathcal{A}. \tag{10}$$

The objective function (7) can be rewritten as

$$\begin{aligned}&\min \sum_{a\in\mathcal{A}} c_a y_a - \sum_{i\in V\setminus\{0\}} \lambda_i \sum_{a\in\mathcal{A}^i} y_a + \sum_{i\in V\setminus\{0\}} \lambda_i = \\ &\min \sum_{a\in\mathcal{A}} (c_a - \lambda_{\ell_a}) y_a + \sum_{i\in V\setminus\{0\}} \lambda_i.\end{aligned}$$

As a consequence, for fixed λ, the Lagrangian relaxation can be solved as a (continuous) minimum-cost network flow problem over the decision diagram, using $c_a - \lambda_{\ell_a}$ as the cost for arc $a \in \mathcal{A}$, yielding an integer optimal solution. In fact, given constraints (9) and the unit capacity constraints on the arcs, each solution consists of K arc-disjoint r-t paths. By applying the successive shortest paths (SSP) algorithm [1] to solve $L(D,\lambda)$ we obtain the following result:

Lemma 1. *Given a decision diagram $D = (\mathcal{N},\mathcal{A})$ and fixed λ, the Lagrangian relaxation $L(D,\lambda)$ can be solved in $O(K(|\mathcal{N}|\log(|\mathcal{N}|) + |\mathcal{A}|))$ time.*

We also implemented a dedicated algorithm, based on the 'minimum update Successive Shortest Paths' (muSSP) algorithm that was developed for specific directed acyclic graphs in the content of multi-object tracking in computer vision [34]. Although graphs with a slightly different structure are considered in [34], the algorithm generalizes to our case: weighted directed acyclic graphs with one source (the root), one sink (the terminal), and unit capacities. The muSSP algorithm leverages the fact that most updates to the shortest path tree through Dijkstra's algorithm are not useful, and it aims instead to make minimal updates to the shortest path tree. While it has the same theoretical worst-case time complexity as the SSP, in practice the muSSP algorithm can be an order of magnitude more efficient than the standard SSP algorithm.

The Lagrangian 'dual' subproblem $\max_\lambda L(D,\lambda)$ finds the multipliers that provide the best Lagrangian bound. Because the objective in $L(D,\lambda)$ is concave and piecewise linear, the dual can be solved via a subgradient method. At each iteration k of the subgradient method, one choice for a subgradient that we will use is γ^k such that $\gamma_i^k = (1 - \sum_{a\in\mathcal{A}^i} y_a^k)$ where y_a^k is the solution to $L(D,\lambda^k)$. Then we update the dual multipliers for the next iteration as $\lambda^{k+1} = \lambda^k + \alpha^k\gamma^k$, where we use an estimated Polyak step size α^k [10]. Note that the initial choice of multipliers λ^0 can be important for solving the dual quickly [9].

We remark that the optimal Lagrangian dual bound is equal to the optimal linear programming bound from model $LP(F(D))$, when both apply the same decision diagram. Moreover, when the column elimination process uses model $LP(F(D))$ or $L(D, \lambda)$, its bound at termination is equal to the column generation bound of the set partitioning model (1), assuming that all methods use the same underlying dynamic programming formulation, as was observed in [27]. That is, the decision diagram applies the same dynamic programming formulation in its construction as column generation uses in the pricing problem.

Lastly, we note that in each iteration of the subgradient method for solving the Lagrangian dual the solution can potentially be used to identify and separate conflicts. Similar to [28], we separate these conflicts in batches of size 100, after which we restart the Lagrangian process.

6 Cutting Planes

Results from the literature show that the LP relaxation of the set partitioning formulation for CVRP, solved via column generation, frequently has a 1-4% optimality gap. To further strengthen the LP relaxation several classes of valid inequalities can be added. According to the literature, the most effective are rounded capacity cuts, strengthened comb inequalities, and subset-row cuts [18,12,23]. The first two types of cuts are called *robust* in the column generation literature because they do not affect the runtime of the pricing problem, while the subset-row cuts are not robust. We next show how rounded capacity cuts and strengthened comb inequalities can be implemented in our decision diagram-based model $LP(F(D))$, as well as a generalization of subset-row cuts as a type of clique cut.

Rounded capacity cuts ensure that a subset of locations S is visited by a sufficient number of trucks to meet its aggregate demand. In column generation these cuts can be added to model (1) so long as the underlying routes are stored for each $r \in R$. Let p_r^S be the number of times route r uses an edge between S and $V \backslash S$, and let $k(S) = \lceil \frac{1}{Q} \sum_{i \in S} q_i \rceil$. The cut added to the restricted master problem is $\sum_{r \in R} p_r^S x_r \geq 2k(S)$, and the associated dual variable is added to controls in the dynamic program for the pricing problem that correspond to a route traversing an edge between S and $V \backslash S$. To add this cut in column elimination, let A^S be the set of arcs $a \in A$ such that $\ell(a) \in S$ and the node u that is the head of a has state with last visited location $i \in V \backslash S$, or the other way around with $\ell(a) \in V \backslash S$ and $i \in S$. A rounded capacity cut for set S can be modeled by adding to $LP(F(D))$ the following inequality: $\sum_{a \in A^S} y_a \geq 2k(S)$. Note that when solving $LP(F(D))$ using the Lagrangean formulation, this constraint can be dualized.

Strengthened comb inequalities are a generalization of comb inequalities that have been proven highly useful for solving the Traveling Salesman Problem [17]. A strengthened comb inequality is defined by a handle set of locations H and teeth sets of locations T_t for $t \in \{1, ..., T\}$. Let $S(H, T_1, ..., T_T)$ be the appropriately defined right hand side for the inequality [17]. In column generation, this cut also requires storing the underlying routes and can be added to the restricted master problem as $\sum_{r\in R} p_r^H x_r + \sum_{t\in T}\sum_{r\in R} p_r^{T_t} x_r \geq S(H, T_1, ..., T_T)$. The associated dual variable is then added to controls in the dynamic program for the pricing problem that correspond to traversing edges with one endpoint in H or one of T_i and the other endpoint not in that set. In column elimination, a strengthened comb inequality with handle H and teeth T_t can be modeled by adding to $LP(F(D))$ the following inequality: $\sum_{a\in A^H} y_a + \sum_{t\in\{1,...,T\}}\sum_{a\in A^{T_t}} y_a \geq S(H, T_1, ..., T_T)$. This constraint can also be dualized when using the Lagrangean formulation to solve $LP(F(D))$.

Subset row cuts are non-robust cuts that have been successfully applied to the CVRP. In particular, the limited memory subset row cuts are an important part of the success of the column generation method in [23]. Since the decision diagram representation does not have a matrix view of the set of routes, subset row cuts do not directly translate to the $LP(F(D))$ model. However, they can be generalized by a class of clique cuts on a specific conflict graph [3]. These cuts are non-robust and have been used in [4] but not until the problem size has been reduced. The structure of our decision diagram allows column elimination to implement a specific version of these cuts. Let $D = (\mathcal{N}, \mathcal{A})$ be a decision diagram as defined above. The conflict graph $G^C = (\mathcal{N}, A^C)$ is defined on node set $\mathcal{N}$. Its arc set A^C contains all arcs (i, j) such that 1) the set of visited locations in the states associated to nodes i and j have a non-empty intersection, and 2) nodes i and j never appear on the same directed path in D. A *clique cut* states that the flow through nodes in a clique of G^C must be at most 1:

Theorem 3. *Let $\mathcal{C}$ be a clique in the conflict graph G^C derived from a decision diagram D. The associated* clique cut $\sum_{i\in\mathcal{C}}\sum_{a\in\delta^-(i)} y_a \leq 1$ *is a valid inequality for model $LP(F(D))$.*

Proof. By construction of G^C, each pair of nodes $i, j \in \mathcal{C}$ has at least one common visited location (say u) in their associate states, and there is no directed path between i to j in D. Suppose that for an integral optimal solution we have $\sum_{a\in\delta^-(i)\cup\delta^-(j)} y_a > 1$. This means that location u is visited twice, which cannot occur in an optimal solution: a contradiction. $\square$

Given G^C and a set of cliques in G^C, clique cuts can be easily separated for $LP(F(D))$ by evaluating whether a given fractional solution violates a cut. Because a solution to the Lagrangian model $L(D, \lambda)$ is integral, we cannot directly use it to separate any cuts. In [2] it is shown that a weighted average of the subproblem solutions converges to an optimal primal solution and we apply this method to identify valid inequalities.

7 Reduced Cost-Based Arc Fixing

Variable fixing based on reduced costs is often applied to reduce the problem size of integer programs [21], including the CVRP [14,23]. It uses a feasible dual solution and suitably small optimality gap to set the value of a primal variable equal to 0 [1,11,16]. We develop an arc fixing method for the $LP(F(D))$ model, using similar arguments as [23].

Let $D = (\mathcal{N}, \mathcal{A})$ be a decision diagram that is exact w.r.t some set of routes $R' \subseteq R$. Consider a feasible dual solution (ν, κ) to the LP relaxation of the set partitioning model (1) over R', where ν correspond to the 'set partitioning' constraints and κ to the 'number of trucks' constraint. For each arc $a \in \mathcal{A}$ we define a 'reduced cost distance' $rc(a) = l_a - \nu_{\ell_a}$. For each node $u \in \mathcal{N}$, we define $sp_u^{\downarrow}$ as the shortest r-u path in D with respect to the reduced cost distances, and similarly define $sp_u^{\uparrow}$ to be the shortest u-t path in D.

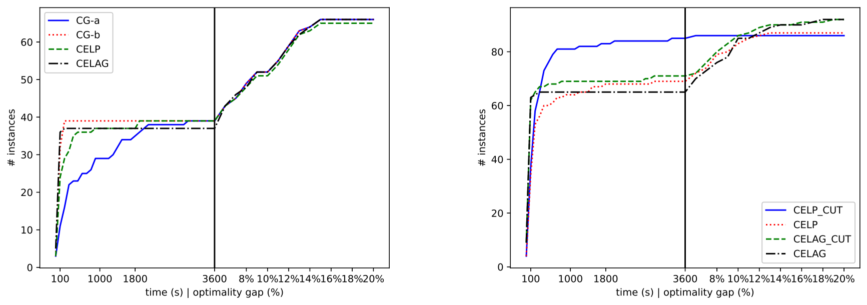
Theorem 4. *Consider arc $a = (v_1, v_2) \in \mathcal{A}$. Let $v(\nu, \kappa)$ be the dual solution value, and let UB an upper bound on (1). If $v(\nu,\kappa) + sp_{v_1}^{\downarrow} + sp_{v_2}^{\uparrow} + rc(a) - \kappa > UB$, then arc a can be fixed to have flow 0 in $F(D)$ and accordingly in $LP(F(D))$.*

Proof. Given (ν, κ), each route $\overline{r} \in R'$ in the LP relaxation of (1) has reduced cost $rc(\overline{r}) = d_{\overline{r}} - \sum_{i=1}^{n} M_{i\overline{r}}\nu_i - \kappa$. Each $\overline{r}$ corresponds to a path $p = \{a_1, ..., a_l\}$ in D, so $rc(\overline{r})$ can be decomposed into $rc(\overline{r}) = \sum_{i=1}^{l} rc(a_i) - \kappa$. For all p that contain arc a, let p' be the path that corresponds to the route r' with lowest reduced cost. Denote $rc(r') = sp_{v_1}^{\downarrow} + sp_{v_2}^{\uparrow} + rc(a) - \kappa$. Now for sake of contradiction assume an optimal solution to $F(D)$ has $y_a = 1$. Then some path p'' in D that contains arc a will have flow of 1, so we can consider this as some $x_{r''} = 1$ in an optimal solution to (1). To construct the remainder of an optimal solution to the LP relaxation of (1) we can solve this LP relaxation with constraints for locations in r'' removed and only requiring $K-1$ trucks. Because (ν, κ) remains feasible to the dual of this updated problem and has value $v(\nu, \kappa) - \sum_{i=1}^{n} M_{ir''}\nu_i - \kappa$, it gives a valid lower bound on (1) that contradicts UB, namely $v(\nu,\kappa) - \sum_{i=1}^{n} M_{ir''}\nu_i - \kappa + d_{r''} = v(\nu,\kappa) + rc(r'') \geq v(\nu,\kappa) + rc(r') \geq v(\nu,\kappa) + sp_{v_1}^{\downarrow} + sp_{v_2}^{\uparrow} + rc(a) - \kappa > UB$. $\square$

Note that while Theorem 4 relies on the set partitioning model (1) to build the reduced cost argument, we can use the optimal dual solution to $LP(F(D))$ in the application of the theorem. When solving $LP(F(D))$ with a standard linear programming solver, we can use the feasible dual from the previous iteration – which remains feasible even with cuts and separations – to fix arcs. One important note is that these fixed arcs are reintroduced if separation happens before the next iteration, as the change in the decision diagram structure may disrupt previous arc fixing arguments. When solving $LP(F(D))$ via its Lagrangian relaxation $L(D, \lambda)$, we must ensure that we work with a feasible dual solution. In addition, we include a dual variable for constraint (10) and set it to its maximum value while ensuring dual feasibility.

8 Experimental Results

We use the benchmark set of CVRP instances from `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/`, including the new challenge set of instances from [30]. All experiments are run on an Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz. We use CPLEX version 22.1 [19] as a linear programming solver



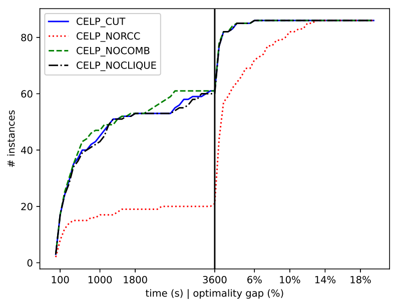
a. Baseline CE vs. CG b. Adding Cuts to CE

Fig. 4. a) Comparing column generation over DP_{Q_2} with column elimination starting from DP_{Q_1} and using different methods to solve $LP(F(D))$ without any added cuts. b) Performance plot for adding cuts to CELP and CELAG.

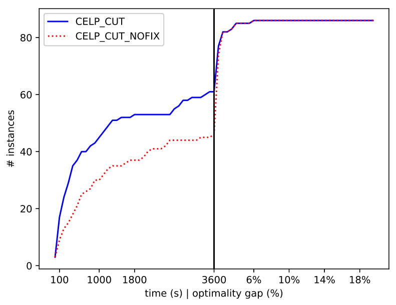
and change 4 to $\geq$ to help find an initial feasible solution. We use the package CVRPSEP [18] to heuristically find rounded capacity cuts and strengthened comb inequalities when given a fractional primal solution. At each iteration we add at most 10 robust capacity cuts and 5 strengthened comb inequalities, using the most violated ones possible. We use Cliquer [22] to heuristically find large weighted cliques in the conflict graph used to derive clique cuts.

Comparing Column Elimination and Column Generation. We compare column generation over DP_{Q_2} with column elimination starting from the DP_{Q_1} route relaxation and eliminating cycles of size 2. Doing so, the final bounds are the same, which allows us to compare how quickly column generation and column elimination reach the optimal bound. We implement a vanilla version of column generation that starts with a small set of greedily chosen routes and solves the pricing problem as shortest paths through the pre-compiled decision diagram for DP_{Q_2}. We compare column generation not including and including time to compile the decision diagram (CG-a, CG-b), column elimination using CPLEX (CELP), and column elimination using a subgradient method over the Lagrangian dual (CELAG). We run each method for 3,600 seconds over benchmark sets A, B, E, F, M, P. We remove instances when the decision diagram for DP_{Q_2} does not finish compiling. Arc fixing uses the best known solution as an upper bound and is used in CELP but not in CELAG. Lower bounds for column generation are computed before termination as in [35]. Figure 4.a is a performance plot of the number of instances solved to within a 5% optimality gap in a given amount of time, extended by the number of instances solved to larger optimality gaps. Over the given relaxation, it is evident that column elimination with the different methods can work appropriately and be competitive with column generation.

Evaluating the Impact of Cuts. We compare solving column elimination using CPLEX with and without cuts (CELP_CUT, CELP) and using the La-



a. Evaluating Individual Cuts



b. Evaluating Arc Fixing

Fig. 5. a) Evaluating the performance of individual cuts on CELP_CUT. b) Performance plot of CELP_CUT with and without arc fixing

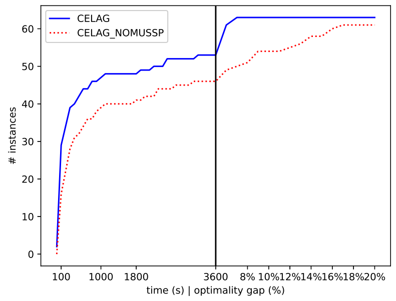
grangian method with and without cuts (CELAG_CUT, CELAG). Figure 4.b is a performance plot for solving the instances up to 5% as in the last experiment. Figure 4.b shows how cuts greatly improve column elimination when using CPLEX as the LP solver, and benefit when solving the Lagrangian reformulation but not as much.

We then compare the performance of CELP_CUT removing one class of cuts at a time: without the rounded capacity cuts (CELP_NORCC), without the strengthened comb inequalities (CELP_NOCOMB), and without the clique inequalities (CELP_NOCLIQUE). Figure 5.a is a performance plot of the number of instances solved to within a 1% optimality gap. Rounded capacity cuts provide the most benefit, the overhead of strengthened comb inequalities sometimes outweigh their benefit but not entirely if we more closely examine the bounds achieved for each instance, and clique inequalities can provide some benefit later in the method when it is able to be distinguished from separations and other cuts.

Evaluating the Impact of Arc Fixing. We consider the impact of arc fixing by removing the feature from CELP_CUT to get CELP_CUT_NOFIX. Figure 5.b is a performance plot using 1% optimality gap. Arc fixing speeds up column elimination to find stronger bounds in less time.

Evaluating the Impact of muSSP. We evaluate the impact of using the muSSP algorithm to solve the subproblem in CELAG by removing it in CELAG_-NOMUSSP. The performance plot using 5% optimality gap is for 64 large X instances and shows that there is a significant speedup. We chose to use the X class here because the speedup is more pronounced on large instances.

Comparison to State-of-the-Art. Figure 6.b compares the state-of-the-art BCP method's root node lower bounds (Pecin) with the best column elimination method settings that we chose through experimentation (CE). For each class of problems the table gives the number of problems in the class (NP) and the average optimality gap found at the root node over all instances. Pecin takes



a. Evaluating muSSP for CELAG

Class	NP	Pecin Gap (%)	CE Gap (%)
A	22	0.36	0.66
B	20	0.14	0.61
E-M	12	0.33	2.60
F	3	0.00	16.41
P	24	0.42	0.85
X	100	0.44	2.13

b. Comparing root node bounds

Fig. 6. (a) A comparison of column elimination using the Lagrangean reformulation with and without the muSSP algorithm. (b) Comparing the root node lower bounds from Pecin et al. [23] (Pecin) and the lower bounds from column elimination (CE); both methods include cuts.

less than 3600 seconds to compute its bounds for all instances except the X class where it can take several hours. CE gaps are computed based on 3600 second runs for all classes except M, F, and X which are given 7200 seconds. The decision diagram did not compile for 12 X instances, so we leave these out of the analysis. One F instance with large capacity resulted in a large diagram and 27% gap that can be reduced with more runtime. The better of the CELAG and CELP results is used; most small instances use CELP while large instances like almost all of the X class use CELAG. We also remove two E class instances with unconventional demand formatting.

9 Conclusion

We introduced a column elimination procedure for the capacitated vehicle routing problem (CVRP). Our methods works with a relaxed set of routes that are compactly represented in a decision diagram, and from which infeasible routes are iterative removed. We showed how we can use existing route relaxations for the CVRP, such as the q-route and ng-route relaxation, to compile good initial relaxed decision diagrams. When the decision diagram is exact, and only contains all feasible routes, we showed that a solution to the CVRP can be found by solving a constrained network flow problem over the diagram. When the diagram is relaxed, this model yields a dual bound. To strengthen the linear programming relaxation of our model we added valid inequalities; in particular, we showed how a class of clique cuts can be derived from the structure of the diagram. To solve the model more efficiently, we considered solving a Lagrangian dual formulation for which we implemented a specialized successive shortest paths algorithm. In our experimental results, we demonstrated that column elimination is a viable alternative to column generation for the CVRP, although the best known dual bounds from the literature, obtained by column generation with cutting planes, are generally stronger.

Acknowledgements

This work is partially supported by Office of Naval Research Grant No. N00014-21-1-2240 and National Science Foundation Award #1918102. This material is also based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1745016, DGE2140739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.
2. Kurt M Anstreicher and Laurence A Wolsey. Two "well-known" properties of subgradient optimization. *Mathematical Programming*, 120(1):213–220, 2009.
3. Egon Balas and Andrew Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. In *Combinatorial Optimization*, pages 37–60. Springer, 1980.
4. Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.
5. Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, 59(5):1269–1283, 2011.
6. M. L. Balinski and R. E. Quandt. On an Integer Program for a Delivery Problem. *Operations Research*, 12(2):300–304, 1964.
7. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, 46(3):316–329, 1998.
8. David Bergman, Andre A Cire, Willem-Jan Van Hoeve, and John Hooker. *Decision diagrams for optimization*, volume 1. Springer, 2016.
9. Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.
10. Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *Lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.
11. Matteo Fischetti and Paolo Toth. An additive bounding procedure for combinatorial optimization problems. *Operations Research*, 37(2):319–328, 1989.
12. Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.
13. S. Irnich and G. Desaulniers. Shortest Path Problems with Resource Constraints. In *Column Generation*, pages 33–65. Springer, 2005.

14. Stefan Irnich, Guy Desaulniers, Jacques Desrosiers, and Ahmed Hadjar. Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, 22(2):297–313, 2010.
15. Anthony Karahalios and Willem-Jan van Hoeve. Variable ordering for decision diagrams: A portfolio approach. *Constraints*, 27(1):116–133, 2022.
16. Andrea Lodi, Michela Milano, and Louis-Martin Rousseau. Discrepancy-based additive bounding for the alldifferent constraint. In *International Conference on Principles and Practice of Constraint Programming*, pages 510–524. Springer, 2003.
17. J. Lysgaard, A. Letchford, and R. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, Series A*, 100:423–445, 2004.
18. Jens Lysgaard. Cvrpsep: A package of separation routines for the capacitated vehicle routing problem. *http://www. hha. dk/˜ lys/CVRPSEP. htm*, 2003.
19. CPLEX User's Manual. Ibm ilog cplex optimization studio. *Version*, 12(1987-2018):1, 1987.
20. Christofides N., Mingozzi A., and Toth P. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282, 1981.
21. G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.
22. Patric RJ Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1-3):197–207, 2002.
23. Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100, 2017.
24. Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.
25. Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, 183(1):483–523, 2020.
26. Roberto Roberti and Aristide Mingozzi. Dynamic ng-path relaxation for the delivery man problem. *Transportation Science*, 48(3):413–424, 2014.
27. Z. Tang and W.-J. van Hoeve. Dual Bounds from Decision Diagram-Based Route Relaxations: An Application to Truck-Drone Routing. *Optimization Online*, 2022.
28. Ziye Tang. *Theoretical and Computational Methods for Network Design and Routing*. PhD thesis, Carnegie Mellon University, 2021.
29. P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications*. SIAM, second edition edition, 2014.
30. Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.
31. W.-J. van Hoeve. Graph Coloring Lower Bounds from Decision Diagrams. In *Proceedings of IPCO*, volume 12125 of *LNCS*, pages 405–419. Springer, 2020.
32. W.-J. van Hoeve and Z. Tang. Column "Elimination": Dual Bounds From Decision Diagram-based Route Relaxations. In *INFORMS Computing Society Conference*, 2022.
33. Willem-Jan van Hoeve. Graph coloring with decision diagrams. *Mathematical Programming*, 192(1):631–674, 2022.

34. Congchao Wang, Yizhi Wang, Yinxue Wang, Chiung-Ting Wu, and Guoqiang Yu. mussp: Efficient min-cost flow algorithm for multi-object tracking. *Advances in Neural Information Processing Systems*, 32, 2019.
35. Laurence A Wolsey. *Integer programming.* John Wiley & Sons, 2020.