Branch-and-Price for Probabilistic Vehicle Routing

Felipe Lagos\(^1\), Mathias A. Klapp\(^2\), Alejandro Toriello\(^1\)

\(^1\)H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

\(^2\)Engineering School
Pontificia Universidad Católica de Chile
Santiago, Chile

felipe.lagos@gatech.edu, maklapp@ing.puc.cl, atoriello@isye.gatech.edu

December 5, 2017

Abstract

The Vehicle Routing Problem with Probabilistic Customers (VRP-PC) is a fundamental building block within the broad family of \textit{a priori} routing models and has two decision stages. In the first stage, the dispatcher determines a set of vehicle routes serving all potential customer locations before the actual requests for service realize. In the second stage, vehicles are dispatched after observing the subset of customers requiring service; a customer not requiring service is skipped from its planned route at execution. The objective is to minimize the expected vehicle travel cost assuming known customer realization probabilities.

We propose a column generation framework to solve the VRP-PC to a given optimality tolerance; as with other branch-and-price VRP schemes, our framework can handle sequence-dependent constraints such as time windows. Specifically, we present two novel algorithms, one that under-approximates a solution’s expected cost, and another that uses its exact expected cost. Each algorithm is equipped with a route pricing mechanism that iteratively improves the approximation precision of a route’s reduced cost; this produces fast route insertions at the start of the algorithm and reaches termination conditions at the end of the execution. We provide \textit{a priori} and \textit{a posteriori} performance guarantees for these algorithms and test their performance on VRP-PC instances with time windows. Our results suggest that both algorithms can numerically optimize instances with up to 40 customers and realization probabilities of 0.5, 0.7 and 0.9.

1 Introduction

The Vehicle Routing Problem (VRP) and its variants are widely studied within the operations research and transportation science communities [13, 23, 37]. Stochastic VRPs are extensions of these problems where a fraction of all relevant instance parameters is unknown while planning and/or executing a solution. \textit{A priori} VRPs are particular versions of stochastic VRPs modeled as
two-stage stochastic integer programs; see [11] for a survey. In the first stage, the decision maker plans an initial solution, where some of the parameters are random variables. In the second stage, the planned solution is executed after a realization of the parameters. Typically, a simple and preestablished recourse rule is used to modify the initial plan during operation. A desired feature in the first stage solution is to proactively account for the second-stage recourse.

We study a novel exact approach for the *a priori* VRP with probabilistic customers (VRP-PC). In this problem, the subset of customers requiring service is random and follows a known probability model. In the first stage, the decision maker plans a set of vehicle routes dispatched from the depot and visiting all potential customer locations. Vehicles are dispatched in the second stage after observing the subset of customers requiring service. The second-stage recourse rule modifies the first-stage routes by skipping locations without a service requirement, maintaining the established sequence of each route for the visited customers. The objective is to minimize the expected vehicle travel cost, accounting for this recourse rule.

The VRP-PC is a fundamental building block to solve more complex dynamic and stochastic VRPs; see *e.g.*, [3, 28, 32, 39], and it is also a natural solution for problem settings where it is infeasible or impractical to plan optimal routes *a posteriori* [7]; this might happen when the decision maker doesn’t have enough resources to post-optimize routes [2], or when maintaining the initial visit sequence is desirable or required, *i.e.*, because of gains in operational efficiency produced by drivers executing the same plan every day, or because customers expect to meet the same driver at roughly the same time every day.

The single-route version of the VRP-PC is known as the Probabilistic Traveling Salesman Problem (P-TSP). An exact algorithm for the P-TSP is proposed in [31], which is a specialized implementation of the integer L-shape method [30]. This method formulates the P-TSP as a two-stage integer linear program with edge-based variables, and replaces the expected route cost in the objective by a variable $\theta \geq 0$. As the solver finds feasible integer solutions, the real expected cost of these solutions is evaluated to dynamically generate “optimality cuts” on $\theta$ and correct its value.

The integer L-shape method operates on a formulation with edge variables. However, many practical sequence-dependent constraints, such as time windows and precedence constraints, are difficult to model with edge variables, and often have weak relaxations. Therefore, to our knowledge the work on the VRP-PC considering such constraints has been restricted to heuristics; see [10,
 Nonetheless, in deterministic problems, route-based formulations solved via branch-and-price (B&P) are arguably more effective to optimize routing models with such constraints (see e.g., [14]). We address this gap in the literature and propose a B&P framework to solve the VRP-PC. In particular, our contributions are:

1. We present two different and independent column generation algorithms, one that underestimates a feasible solution’s cost and another that uses its exact expected cost. Both algorithms use route generation subroutines that compute estimates of a route’s reduced cost. The algorithms’ pricing problems iteratively increase the precision of a route’s expected cost estimate, yielding fast route generation at the start while still reaching termination conditions at the end of the execution.

2. We provide *a priori* and *a posteriori* performance guarantees for these algorithms, which allow us to determine solution quality before and after a lower bound on the optimal value is computed.

3. We test the performance of our algorithms on computationally simulated instances of VRP-PC with time windows, and conclude that both find good solutions using only a few number of precision updates. The algorithm’s empirical convergence takes few iterations, and depends on the customer realization probabilities.

The remainder of this article is organized as follows. Section 1.1 provides a literature review. Section 2 reviews the B&P approach for the VRP and formulates the VRP-PC. In Section 3 we present two column generation algorithms for the VRP-PC, discuss incorporating them into B&P, and give convergence guarantees and approximate optimality conditions. Section 4 presents a computational study on modified Solomon instances [34], and Section 5 concludes.

1.1 Literature Review

*A priori* optimization is routinely applied in stochastic routing problems [8, 27, 36]. Different sets of uncertain parameters within VRPs are modeled through different *a priori* optimization problems. For example, the VRP with stochastic travel times (VRP-STT) refers to uncertainty in travel times between locations [29, 33] and the VRP with stochastic demand (VRP-SD) [5, 7] refers to a VRP where the customer demand is a random variable realized at the moment of service.
The *a priori* VRP with probabilistic customers (VRP-PC) is a stochastic VRP where the subset of customers requiring service is random and follows a known probability distribution. [2] is an early work in this vein, which solves a food delivery problem by designing *a priori* routes based on TSP heuristics. The seminal work in [25, 26] formally introduces the Probabilistic Traveling Salesman (PTSP) and develops a closed-form expression for a route’s expected cost in the case with homogeneous probabilities. [4] extends the PTSP to heterogeneous probability distributions, and additional results include [6, 7, 8, 27].

[30] present the integer L-shape method to exactly solve *a priori* optimization problems; the algorithm gradually improves an estimate of the expected cost via dynamic generation of cutting planes in an integer program. [31] use the integer L-shape method to solve an edge-based formulation of the PTSP. This method is also implemented in [21] to design an exact algorithm for a VRP with both probabilistic customers and stochastic demand.

Truncation approaches have been proposed by [22] for the PTSP; they underestimate the true cost by computing only some terms of the expected cost of a route. [35] underestimate the true cost using an approximation function; this function balances the amount of speedup for solving algorithms and the quality of the approximation. Such approximation functions have been used to speed up heuristics [9].

For deterministic routing, B&P approaches have been successfully applied by formulating the VRP as a set covering route-based formulation; see e.g. [1, 15, 16] and references therein. Recently, B&P has been adapted to solve the VRP-SD [12, 17, 20] and the VRP-STT [19].

The route pricing problem within a VRP set partitioning formulation is the Elementary Shortest Path Problem with Resource Constraints (ESSPRC), which is strongly NP-hard. Because of the ESSPRC’s complexity and empirical difficulty, a relaxed version called called the Shortest Path Problem with Resource Constraints (SPPRC) [14] is usually solved instead; the SPPRC relaxes the elementary condition to allow visiting a customer more than once. The SPPRC with *k*-cycle elimination [24] is an intermediate relaxation that only allows paths with cycles having at least \( k + 1 \) nodes.
2 Model Formulation and Preliminaries

2.1 Deterministic VRP

The deterministic VRP entails designing a set of vehicle routes, each starting and ending at a depot, that feasibly serve a set \( C = \{1, \ldots, n\} \) of customers at minimum total travel cost. Each customer \( i \in C \) has demand \( d_i > 0 \) that must be served by one route. We consider homogeneous vehicles with capacity \( q \geq \max_i d_i \), so that the total demand served on any route does not exceed \( q \).

Define the set \( N := C \cup \{0, n+1\} \) of nodes having all customers plus 0 and \( n+1 \), both of which represent the depot. A vehicle route departs node 0, visits a subset of customer nodes, and ends at node \( n+1 \). Traveling between any two nodes \( i, j \in N \) costs \( c_{ij} \geq 0 \) and takes \( t_{ij} \geq 0 \) time, and we assume both parameter sets satisfy the triangle inequality. When \( i \in C \), the value \( t_{ij} \) may include service time at \( i \), and we have a route duration limit \( T \), with \( T \geq t_{ij} \) for all arcs.

We are interested in VRP models that include complex constraints amenable to column generation and B&P. Time windows are among the most commonly studied constraints in VRP B&P models, so we specifically formulate and study the VRP with Time Windows (VRPTW) as a proof of concept for our approach. In this model, the arrival at \( i \in C \) must occur within a time interval \( [a_i, b_i] \subseteq [0, T] \). We assume hard time window constraints; if a vehicle arrives too early at a customer, it must wait until the window’s start time, while late arrivals are not allowed.

2.2 Column Generation and Branch-and-Price

To specify the B&P approach, we assume all applicable parameters are integers, possibly by re-scaling. Define \( R_n \) as the set of all feasible vehicles routes; each route \( r \in R_n \) corresponds to an elementary path of nodes in \( C \) starting at 0, ending at \( n+1 \) and satisfying the problem constraints, including vehicle capacity and time windows. Let \( c_r \) be the cost of route \( r \) and \( \alpha^i_r \in \{0,1\} \) be the number of times \( i \in C \) is visited by \( r \).

Define the binary variable \( y_r \), equal to 1 if route \( r \) is selected and 0 otherwise. The VRPTW can then be stated as a set partitioning integer linear problem, where its linear relaxation is given
by

\[
\min_{y \geq 0} \sum_{r \in R_n} c_r y_r \quad (1a)
\]

s.t. \[
\sum_{r \in R_n} \alpha^i_r y_r = 1, \quad i \in C.
\] (1b)

The number of routes in \( R_n \) can be exponentially large as a function of \( n \), making it difficult to solve (1) explicitly with an LP solver. Instead, routes can be generated dynamically using column generation. To solve the VRPTW set partition LP relaxation, we use an algorithm with two main components. Initially, we solve model (1) only considering a small subset \( \tilde{R} \subset R_n \) of feasible routes. The optimal dual solution of this restricted LP is then used to identify profitable columns in \( R_n \setminus \tilde{R} \) via a pricing subproblem. We add these new columns to \( \tilde{R} \), then execute a new run of the LP solver. The procedure is repeated until no more profitable columns are found in \( R_n \setminus \tilde{R} \).

Let \( \rho_i \) for \( i \in C \) be the dual LP variable related to constraint (1b). The pricing problem to generate routes based on the current restricted LP solution is

\[
\min_{r \in R_n} \left\{ c_r - \sum_{i \in C} \alpha^i_r \rho_i \right\}. \tag{2}
\]

A non-negative optimal value of (2) certifies optimality for the restricted version of (1). Conversely, a negative value indicates the existence of a profitable route \( r \in R_n \setminus \tilde{R} \).

Problem (2) is known as the Elementary Shortest Path Problem with Time Windows and Capacity Constraints (ESPPTWCC), and it is strongly NP-hard [18]. The Shortest Path Problem with Resource Constraints (SPPRC) is a relaxation of (2) that allows multiple visits to a customer; this relaxed version can be solved by dynamic programming (DP) in polynomial time as a function of \( n \), \( q \) and \( T \). A relaxed route \( r \) can have cycles, but cannot make more than \( n \) customer visits, and may in fact be further constrained, since time and capacity resources are increasingly consumed along any path. We consider \( k \)-cycle elimination for the SPPRC (SPPRC-\( k \)) [24] to improve relaxation quality; in this setting any path with cycles having fewer than \( k + 1 \) nodes is not allowed. Let \( R_k \) be the set of relaxed routes with \( k \)-cycle elimination, and observe that since elementary paths exclude all cycles up to length \( n \), this definition is consistent with our use of \( R_n \).

The SPPRC is defined over a network of partial route states \( G = (V, A) \). Each state \( v \in V \)
is specified by a tuple \( v = (S_v, t_v, d_v) \), where \( S_v \) is a sequence of previously visited locations in \( N \), \( t_v \) is the current time of the partial route, and \( d_v \) is the current vehicle capacity used. The ESSPRC carries the complete node visiting sequence from the depot in \( S_v \), while the SPPRC-\( k \) relaxation aggregates state information and only records the latest \( k \) visited nodes in the sequence.

The transition cost between states \( v \) and \( u \) is defined by \( c_{\ell(S_v), \ell(S_u)} - \rho_{\ell(S_v)} \), where \( \ell(S) \) is the last node in sequence \( S \).

The DP optimality equations for the SPPRC-\( k \) are

\[
F([0], 0, 0) = 0, \\
F(S, t, d) = \min_{(i, j) \in N^2} \{ F(\bar{S}, \bar{t}, \bar{d}) + (c_{ij} - \rho_i) ; (i, j) = (\ell(\bar{S}), \ell(S)), i \neq n + 1, f_{ij}(\bar{t}, \bar{d}) \leq (t, d) \},
\]

where \( \rho_0 = 0 \), \( F(S, t, d) \) is the smallest reduced cost of a partial route starting from the depot, ending with sequence \( S \) (where \( |S| \leq k \)), ready to leave at time \( t \) or later from \( \ell(S) \) with a used vehicle capacity \( d \). The relaxed route with minimum reduced cost is given by the minimum over all values \( F(S, T, q) \) with \( \ell(S) = n + 1 \). In the recursion, \( \bar{S} \) is a sequence with \( |\bar{S}| \leq k \) that can be extended to \( S \); that is, either \( |\bar{S}| < k \) and appending \( \ell(S) = j \) to it yields \( S \), or \( |\bar{S}| = k \) and \( S \) consists of deleting the first element and appending \( j \). The function \( f_{ij} \) is called a resource extension function [14], and in this case models time and capacity consumption,

\[
f_{ij}(t, d) = (t + t_{ij}, d + d_j),
\]

but it can be more general and include any other resource consumed along a route. We consider the Bellman-Ford labeling algorithm to solve (3). As the number of states satisfies \( |V| = O(n^k qT) \), the running time of the algorithm is \( O(n^{k+1} qT) \).

The optimal values of (1) may be fractional; however, branching on the \( y \) variables is generally impossible (as the pricing problem cannot be readily updated) and would lead to extremely unbalanced search trees. Instead, for VRP the typical B&P scheme [14] branches on the implied arc variables \( x_{ij} \) that indicate whether any route in the solution travels directly from \( i \) to \( j \). Adjusting the pricing SPPRC-\( k \) model for these branching decisions simply involves forcing or forbidding some actions at certain states. In our implementation, we also initially branch on the number of
2.3 VRP with Probabilistic Customers

We consider a VRP with probabilistic customers (VRP-PC), where an initial solution covering all customers $C$ is planned, but only a subset of customers will actually require service. As a recourse rule, a customer that does not require service is skipped in its corresponding route, while keeping the rest of the route’s sequence; for many VRP constraints of interest (including vehicle capacity and service time windows), this ensures that if the route is feasible for the deterministic problem where all customers are serviced, it remains feasible for any realized subset of customers in the probabilistic context. We assume a known, independent customer realization probability $p_i \in (0, 1]$ for each $i \in C$ and use $p_0 = p_{n+1} = 1$. The objective is to minimize the expected vehicle travel cost.

For a route $r \in R_n$, let $n_r \leq n$ be the number of planned customer visits, and let $r(i) \in N$ represent the $i$-th planned visit in $r$ (with $r(0) = 0$, $r(n_r + 1) = n + 1$). The expected cost $\mathbb{E}(c_r)$ of the route is

$$
\mathbb{E}(c_r) = \sum_{i=0}^{n_r} \sum_{j=i+1}^{n_r+1} \left( p_{r(i)} p_{r(j)} \prod_{\ell=i+1}^{j-1} (1 - p_{r(\ell)}) \right) c_{r(i),r(j)}
= \sum_{\ell=1}^{n_r} \sum_{i=0}^{n_r - \ell + 1} \left( p_{r(i)} p_{r(i+\ell)} \prod_{j=i+1}^{i+\ell-1} (1 - p_{r(j)}) \right) c_{r(i),r(i+\ell)} = \sum_{\ell=1}^{n_r} H^\ell_r.
$$

This expected cost can be computed as a sum of $n_r$ nonnegative terms as in (4), where each $H^\ell_r$ includes the expected costs corresponding to arcs that skip exactly $\ell - 1$ customers. For example, the term $H^1_r = \sum_{i\leq n_r} p_{r(i)} p_{r(i+1)} c_{r(i),r(i+1)}$ considers all arcs between consecutive customers. The term $H^2_r$ includes all arcs that skip one customer in the sequence, and so on; note that $H^{n_r+1}_r = 0$. Since they are all non-negative, a summation of any subset of the $H^\ell_r$ terms provides a lower bound for $\mathbb{E}(c_r)$. The definition of $\mathbb{E}(c_r)$ can be extended to include routes with repeat customer visits, by simply setting $c_{r(i),r(j)} = 0$ when $r(i) = r(j)$ and keeping all other definitions the same.

A set partitioning relaxation for the VRP-PC is

$$
f(R_k) := \min_{y \geq 0} \left\{ \sum_{r \in R_k} \mathbb{E}(c_r) y_r : \sum_{r \in R_k} \alpha^i_r y_r = 1, i \in C \right\}.
$$
Compared to a deterministic VRP relaxation, the feasible region remains unaltered, but the objective function considers each route’s expected cost. As in the deterministic case, we can relax the set of feasible routes from $R_n$ to a larger set $R_k$ that only excludes $k$-cycles; we make the dependence on the set of feasible routes explicit and use $f(R_k)$ to denote the optimal value of this relaxation as a function of the considered route set.

3 Column Generation for VRP-PC

The pricing subproblem for the VRP-PC master problem (5) is

$$\min_{r \in R_k} \left\{ \mathbb{E}(c_r) - \sum_{i \in C} \alpha_i^r \rho_i \right\},$$

where the $\rho_i$ are again dual multipliers. Even though we relax the route set to $R_k$, the expected cost formula (4) depends on the entire customer visit sequence and increases the subproblem’s difficulty.

Define

$$E^k(c_r) := \sum_{\ell=1}^{k} H_r^\ell,$$

a lower bound for $\mathbb{E}(c_r)$ for $k \in \{1, \ldots, n_r\}$, where we include arcs that skip up to $k - 1$ customers in the sequence.

**Proposition 1.** For any route $r \in R_1$ and $k \in \{1, \ldots, n_r\}$, the $k$-term approximation of the expected cost $\mathbb{E}(c_r)$ is monotone non-decreasing in $k$, i.e., $E^k(c_r) \leq E^{k+1}(c_r)$. Moreover, if $\bar{c}_r$ is the deterministic cost of visiting all customers in $r$, $E^{nr}(c_r) = \mathbb{E}(c_r) \leq \bar{c}_r$.

**Proof.** $E^{k+1}(c_r) - E^k(c_r) = H_r^{k+1} \geq 0$, so this approximation is monotone non-decreasing. The exact expected cost considers all non-negative terms $H_r^\ell$ and is thus an upper bound to $E^k(c_r)$. Finally, the deterministic cost is no smaller than the expected value because under the triangle inequality, the cost of each possible realization is always less than or equal to the cost of visiting all customers. \qed
3.1 Updating Cost Algorithm

In our first algorithm, which we call the Updating Cost Algorithm (UCA), we obtain a lower bound for the optimal reduced cost of the relaxed master problem (5) by approximately solving subproblem (6) using a non-elementary path relaxation with \(k\)-cycle elimination (SPPRC-\(k\)), and replacing the exact expected cost \(E(c_r)\) with the approximation \(E^k(c_r)\). When solving the SPPRC-\(k\), we can adapt the DP recursion (3) to include expected arc costs corresponding to arcs that skip at most \(k-1\) customers, thus allowing us to optimize with respect to \(E^k(c_r)\).

We first address how well \(E^k\) approximates the true expected cost. Let \(\hat{p} := \max_{i \in C} p_i\), \(\tilde{p} := \min_{i \in C} p_i\), and let

\[
\hat{c} := \max\left\{ \max_{i,j \in C} c_{ij}, \max_{i \in C} \left\{ \max\{ c_{0i}, c_{i,n+1} \} / p_i \right\} \right\}.
\]

This last quantity represents the most expensive arc cost in the graph, where we weigh arcs incident to the depot more heavily.

**Lemma 2.** Let \(y \geq 0\) satisfy \(\sum_{r \in R_1} \alpha^i_r y_r = 1\) for each \(i \in C\), and let \(k \in \{1, \ldots, n\}\). Then

\[
\sum_{r \in R_1} y_r E^k(c_r) \leq \sum_{r \in R_1} y_r E(c_r) \leq \sum_{r \in R_1} y_r E^k(c_r) + \delta_k,
\]

where

\[
\delta_k := \hat{c} \hat{p}^2 \sum_{\ell=k+1}^{n} (n - \ell + 2)(1 - \tilde{p})^{\ell-1}.
\]

Although we state the result in terms of the largest route set \(R_1\), the same guarantee applies to any smaller set \(R_k\) by taking \(y_r = 0\) for \(r \notin R_k\).

**Proof.** The first inequality is a consequence of Proposition 1. To prove the second, we first note that for any route \(r \in R_1\),

\[
E(c_r) - E^k(c_r) = \sum_{\ell=k+1}^{n_r} H^\ell_r \leq \hat{c} \hat{p}^2 \sum_{\ell=k+1}^{n_r} (n_r - \ell + 2)(1 - \tilde{p})^{\ell-1}.
\]
Summing over the \( y_r \) values, we obtain

\[
\sum_{r \in R_1} y_r (\mathbb{E}(c_r) - E^k(c_r)) \leq \hat{c} \hat{p}^2 \sum_{r \in R_1} y_r \sum_{\ell = k+1}^{n_r} (n_r - \ell + 2)(1 - \hat{p})^{\ell-1}
\]

\[
\leq \hat{c} \hat{p}^2 \sum_{\ell = k+1}^{n} (1 - \hat{p})^{\ell-1} \sum_{r \in R_1} y_r (n_r - \ell + 2) \leq \delta_k,
\]

where the last inequality follows from \( \sum_r n_r y_r = \sum_r \sum_i \alpha_i^r y_r = n \) and \( \sum_r y_r \geq 1 \).

This result allows us to gauge how closely we approximate our problem’s true optimal cost if we use the approximate route cost \( E^k \) instead.

**Theorem 3.** Suppose we replace \( \mathbb{E} \) with \( E^k \) in (5), optimize with respect to this objective, and obtain solution \( y^k \). Then

\[
\sum_{r \in R_k} y_r^k E^k(c_r) \leq f(R_k) \leq \sum_{r \in R_k} y_r^k \mathbb{E}(c_r) \leq \sum_{r \in R_k} y_r^k E^k(c_r) + \delta_k.
\]

The analogous guarantee holds for the integral case: Suppose \( R^* \) is an optimal set of routes for the VRP-PC, and suppose \( \bar{R}^k \) is an optimal set with respect to the approximate objective \( E^k \). Then

\[
\sum_{r \in \bar{R}^k} E^k(c_r) \leq \sum_{r \in \bar{R}^*} \mathbb{E}(c_r) \leq \sum_{r \in \bar{R}^k} \mathbb{E}(c_r) \leq \sum_{r \in \bar{R}^k} E^k(c_r) + \delta_k.
\]

**Proof.** The first inequality is a consequence of Lemma 2 and \( y^k \)'s optimality with respect to \( E^k \). The second follows from \( y^k \)'s feasibility for (5), and the last from Lemma 2. The same argument can be repeated for the second set of inequalities, restricting the analysis to integer \( y \) solutions.

The theorem implies that if we approximately optimize (5) over routes \( R_k \) with objective \( E^k \), we have the *a priori* guarantee that the solution we obtain will be within an additive gap of \( \delta_k \) from \( f(R_k) \). Similarly, if we embed this approximate optimization within a B&P algorithm, we are guaranteed to obtain an integer solution within \( \delta_k \) of the optimal expected cost. In addition, after carrying out the optimization, we can calculate a tighter *a posteriori* gap by taking the difference of the solution’s true expected cost with \( \mathbb{E} \), minus its approximate expected cost as measured by \( E^k \).
Corollary 4. To achieve any desired additive optimality gap $\epsilon \geq 0$ in (5) (and in the integral problem via B&P), it suffices to choose $k = O(\log(n/\epsilon))$.

Proof. By definition of $\delta_k$, we have

$$\delta_k \leq \hat{c}p^2 n \sum_{\ell=k+1}^{n} (1 - \hat{p})^{\ell-1} \leq \frac{\hat{c}p^2 n (1 - \hat{p})^k}{\hat{p}}.$$ 

Therefore, to guarantee $\delta_k \leq \epsilon$, it suffices for $k$ to satisfy

$$(1 - \hat{p})^{-k} \geq \frac{\hat{c}p^2 n}{\epsilon \hat{p}} \iff k \ln \left( \frac{1}{1 - \hat{p}} \right) \geq \ln \left( \frac{\hat{c}p^2 n}{\epsilon \hat{p}} \right).$$

Although this last result shows a logarithmic dependence on $n$, in practice we find that the $\delta_k$ values decrease quite rapidly, so that a small $k$ suffices to provide a very tight gap. Table 1 provides an example of $\delta_k/\hat{c}$ values for $n = 50$ as a function of $k$ and a uniform customer probability. Our computational results detailed in the next section verify this convergence and also explore the \textit{a posteriori} gap in empirical terms.

$$\begin{array}{|c|cc|cc|cc|}
\hline
p \backslash k & 1 & 2 & 3 & 4 & 5 \\
\hline
0.5 & 12.25 & 6.00 & 2.9375 & 1.4375 & 0.703125 \\
0.7 & 10.41 & 3.06 & 0.8991 & 0.26406 & 0.077517 \\
0.9 & 4.49 & 0.44 & 0.0431 & 0.00422 & 0.000413 \\
\hline
\end{array}$$

Table 1: Sample $\delta_k/\hat{c}$ values as a function of $k$ and $p = \hat{p} = \hat{p}$, for $n = 50$. 


Algorithm 1: Column generation algorithm UCA.

Algorithm (1) details our implementation of this approximate optimization. Instead of starting from the desired approximation precision, the algorithm solves the column generation algorithm sequentially, with increasing precision at every step of the outer loop. Intuitively, we expect pricing subproblems with small $k$ to be easy, and thus to quickly find useful columns in the first steps; for larger values of $k$, we then obtain a few remaining columns that marginally improve the expected cost. Because the algorithm updates the cost approximation as it progresses, we name it the Updating Cost Algorithm (UCA).

UCA takes as argument an initial set of feasible routes $\tilde{R}$ and two positive integers $K_0 \leq K$; $K_0$ is the initial value for each route’s expected cost approximation and $K$ is the final value used in the approximation for column generation. In each step $k \in \{K_0, \ldots, K\}$, the algorithm searches for routes $r \in R_k$, approximating $r$’s expected cost using $E^k(c_r)$. If the resulting route $r$ has negative reduced cost, we include it in the set $\tilde{R}$ for the master problem; otherwise, we increase $k$ and recompute the cost of the routes considered in the master using $E^k$. 
UCA begins by generating routes in $R_{K}$, meaning some of these routes may in fact have cycles shorter than or equal to $K$. Therefore, we cannot claim that the algorithm optimizes the LP relaxation (5) over $R_{K}$. We can, however, make a weaker assertion.

**Proposition 5.** The value returned by UCA with parameter $K$ is a lower bound for $f(R_{K})$.

**Proof.** In its final iteration ($k = K$), the algorithm ensures that every route in $R_{K}$ has non-negative reduced cost with respect to $E^{K}$. However, the algorithm can generate routes in its previous iterations, some of which could have cycles of length $K$ or shorter. So UCA produces a solution that is optimal for a superset of $R_{K}$, where the inclusion may be strict. 

Finally, we verify that employing UCA within a B&P framework preserves the gap guarantee.

**Corollary 6.** Using UCA within a B&P algorithm yields an integer solution with expected cost within an additive gap $\delta_{K}$ of optimal.

**Proof.** The proof follows from Theorem 3 and Proposition 5 by noting that if UCA returns an integer solution, this solution must be optimal with respect to $E^{K}$. 

We finish this subsection by noting that although UCA has both a priori and a posteriori guarantees, it cannot guarantee a solution with lower expected cost than the solution implied by the deterministic VRP on the same instance. Therefore, in practice we warm start the algorithm with the deterministic solution, which also helps us fathom nodes in the search tree.

### 3.2 Fixed Cost Algorithm

The motivation for UCA and the use of the cost approximation $E^{k}$ is the difficulty of the exact pricing problem (6). However, once we generate a particular route $r$, we can efficiently check its exact expected cost and thus its exact reduced cost. This motivates a second algorithm to approximately solve (5) and the VRP-PC, in which we include routes in the restricted master problem with their exact expected costs; because this second algorithm always keeps routes’ true expected cost (instead of updating an approximation), we call it the Fixed Cost Algorithm (FCA).

Algorithm 2 details FCA. We again use two parameters $K_{0} = K$ and increase the precision of the pricing problem and the expected cost approximation $E^{k}$ for $k \in \{K_{0}, \ldots, K\}$; however, in
this case we only add routes if their exact reduced cost (measured with the exact expected cost) is negative. This has the benefit of including columns in the master with their exact objective value, but the disadvantage that we may have an inconclusive pricing outcome, where a route with negative approximate reduced cost in fact has non-negative reduced cost. Such an inconclusive outcome triggers an increase in the pricing precision until we reach $K$, at which point the algorithm terminates.

Algorithm 2: Column generation algorithm FCA.

**Theorem 7.** The non-basic routes in the solution produced by algorithm FCA have reduced cost bounded below by $-\delta_K$.

**Proof.** If the algorithm terminates because the minimum approximate reduced cost is non-negative, the current solution is optimal. Therefore, assume the algorithm terminates because the final route $\tilde{r}$ priced by the algorithm has an inconclusive reduced cost. That is,

$$E^K(c_{\tilde{r}}) - \sum_{i \in C} \alpha^i_{\tilde{r}} \rho_i < 0 \leq E(c_{\tilde{r}}) - \sum_{i \in C} \alpha^i_\tilde{r} \rho_i,$$
where $\rho$ is an optimal dual solution of the restricted master solved with the current route set $\tilde{R}$. This implies for any route $r \in R_K$ that

$$
\sum_{i \in C} \alpha_i^r \rho_i - \mathbb{E}(c_r) \leq \sum_{i \in C} \alpha_i^r \rho_i - E^K(c_r) \leq \sum_{i \in C} \alpha_i^\tilde{r} \rho_i - E^K(c_{\tilde{r}}) \leq \mathbb{E}(c_{\tilde{r}}) - E^K(c_{\tilde{r}}) = \sum_{t=K+1}^{n_t-1} H_t^\tilde{r} \leq \delta_K,
$$

where the second inequality follows because $\tilde{r}$ is the route produced by the approximate pricing problem.

Like UCA, this algorithm has an \textit{a priori} gap guarantee.

\textbf{Corollary 8.} Suppose $K_0 = K$ and let $y^*$ be optimal for (5) with respect to $R_K$. The solution produced by FCA has objective value no greater than

$$
\sum_{r \in R_K} y^*_r \mathbb{E}(c_r) + \delta_K = f(R_K) + \delta_K \sum_{r \in R_K} y^*_r.
$$

As with UCA, a similar guarantee applies when $K_0 < K$, except the set of routes we optimize over may be larger than $R_K$, as it contains any routes with smaller cycles that the algorithm included in earlier iterations.

\textbf{Corollary 9.} Using FCA within a B&P algorithm yields an integer solution with expected cost within an additive gap $\delta_K n$ of optimal. The factor of $n$ can be substituted by any known tighter bound on the number of routes used by an optimal solution.

\textbf{Proof.} The gap given in Corollary 8 depends on an optimal solution of the LP; in B&P, this solution would vary by node, so we can only claim an overall gap that is guaranteed no smaller than the gap at any node. Since we may assume $\sum y_r \leq n$ without loss of optimality, the result follows. If we have an upper bound on $\sum y_r$ that is tighter than $n$, the same argument applies with this bound.

\textbf{Corollary 10.} To achieve any desired additive optimality gap $\epsilon \geq 0$ for the VRP-PC via FCA within B&P, it suffices to take $K = O(\log(n/\epsilon))$.

\textbf{Proof.} The proof is identical to Corollary 4, except we start with $\delta_K n \leq \epsilon$.  

16
We test our algorithms on VRP-PC instances based on the Solomon VRPTW instances, see e.g., [34]. The Solomon instances have 100 customers each and are divided into three categories: C (clustered), R (uniformly distributed) and RC (mix of R and C). Using 5 instances each from type C (C101 to C105) and 5 from R (R101 to R105), we create VRP-PC instances with 15, 25 and 40 customers. Instances with a given number of customers have a different sequence of customers with respect to the original Solomon instance. For example, we create two 40-customer instances from each original Solomon instance, one considering customers 1 to 40 and another with customers 41 to 80. Because our largest instances have 40 customers, we reduce the vehicle capacity from 200 to 80. All instances have the original depot location.

With this procedure, we respectively obtain 60, 40 and 20 “base” deterministic VRPTW instances with 15, 25 and 40 customers. We then tested our B&P implementation on each of these deterministic instances, eliminating two 25-customer and four 40-customer instances we could not solve to optimality within a 6-hour time limit; this reduction allows us to compare our VRP-PC results against the corresponding deterministic solution, and also lets us focus more on the difficulty increase brought on by the problem’s probabilistic nature, rather than on the challenges it inherits from the VRPTW. After this elimination, each remaining base deterministic instance generates three VRP-PC instances, each with a different uniform customer probability $p \in \{0.5, 0.7, 0.9\}$, yielding a total of 342.

We tested both algorithms, UCA and FCA, on each instance with $K_0 = 1$ and $K \in \{1, \ldots, 5\}$, applying the 6-hour time limit. In total, this involves 1,710 runs of each algorithm on the different instances. We ran the experiments in the Georgia Tech ISyE computing cluster, which uses HTCondor to manage its jobs, on an Intel Xeon E5-2603 (1.80GHz) machine with up to 10Gb of RAM, and using CPLEX 12.4 as LP solver.

In our first set of results, shown in Table 2, we report the average maximum value of $K$ our B&P algorithms were able to solve to optimality and within a 5% and 10% relative gap. (The gap measured here is between the best integer solution and best bound found by the B&P tree.) For example, for instances with 15 customers and probability 0.5, the average largest $K$ value for which our UCA B&P algorithm can report a 0% relative gap within the time limit is 4.45. The
corresponding average for FCA is 4.52.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>UCA</th>
<th></th>
<th></th>
<th>FCA</th>
<th></th>
<th></th>
<th>Total Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0%</td>
<td>5%</td>
<td>10%</td>
<td>0%</td>
<td>5%</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.5</td>
<td>4.45</td>
<td>4.83</td>
<td>4.95</td>
<td>4.52</td>
<td>4.88</td>
<td>4.95</td>
<td>900</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>4.67</td>
<td>4.9</td>
<td>4.97</td>
<td>4.78</td>
<td>4.83</td>
<td>4.98</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>4.87</td>
<td>4.9</td>
<td>5</td>
<td>4.87</td>
<td>4.93</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.5</td>
<td>3.37</td>
<td>4.08</td>
<td>4.68</td>
<td>3.55</td>
<td>4.29</td>
<td>4.71</td>
<td>570</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>3.45</td>
<td>4.11</td>
<td>4.74</td>
<td>3.58</td>
<td>4.16</td>
<td>4.76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>3.71</td>
<td>4.11</td>
<td>4.58</td>
<td>3.68</td>
<td>4.37</td>
<td>4.71</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0.5</td>
<td>2.5</td>
<td>3.19</td>
<td>4.38</td>
<td>3</td>
<td>3.94</td>
<td>4.19</td>
<td>240</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>2.38</td>
<td>3.69</td>
<td>4.25</td>
<td>2.62</td>
<td>3.88</td>
<td>4.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>2.62</td>
<td>4.38</td>
<td>4.5</td>
<td>2.56</td>
<td>4.19</td>
<td>4.44</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Average largest $K$ solved by each algorithm within a certain relative gap.

From the table we see the clear impact the number of customers $n$ has on the parameter $K$ we can use when running the algorithms to optimality. For 15 customers, $K$ can be 4 or 5; for 25 customers, $K$ can be about 3 or 4; and for 40 customers $K$ can be 3 and sometimes 2. Similarly, the customer realization probability $p$ affects this average $K$; as we might expect, the larger the probability, the larger $K$ can be, though there are some exceptions. The choice of $K$ has two separate consequences. First, it helps determine the $a$ priori and $a$ posteriori additive gap guarantees of solution quality we get from UCA and FCA, since they depend on the number of terms we consider in the expected value approximation $E^K$. Second, it impacts the problem’s difficulty through the pricing problem, which is solved as an SPPRC-$K$.

In Figure 1 we depict the average UCA $a$ priori and $a$ posteriori gap guarantees as a function of $K$ for the three different customer probabilities $p \in \{0.5, 0.7, 0.9\}$. For each value of $K$ and $p$, we include in the average only those instances we were able to solve to optimality in the B&P algorithm. We report these gaps as relative distances to the optimal solution; the gaps decay quite significantly for higher $K$ values, so we present them in natural logarithmic scale (i.e., as powers of $e$). For example, for $p = 0.5$ and $K = 3$ the average $a$ priori gap is roughly $1/e \approx 37\%$, meaning we can guarantee before running the algorithm that the expected cost of the solution returned by UCA is at worst roughly a third costlier than the optimum, on average.

These results indicate how quickly both guarantees converge to zero as we increase $K$. For any probability, $K = 4$ or $K = 5$ more than suffice for either algorithm to return a solution with expected cost very close to optimal. Furthermore, for $K \geq 3$ the difference between $a$ priori and
Figure 1: Average a priori and a posteriori guarantees by probability.

a posteriori gaps is already within 10%-20% or less, with the a priori guarantee at worst being about 30% from optimal.

Figure 2 shows the convergence between the UCA and FCA solutions’ expected cost and UCA’s bound in an absolute scale, as a function of the algorithm parameter $K$, plotted by customer realization probability. The averages here include only those instances for which we were able to run the B&P algorithm to optimality for all values of $K$, in order to make the comparison using a fixed set of instances. The figure plots the average of the the UCA optimal value measured with approximation $E^K$, which is a lower bound on the optimal expected cost; it also plots the UCA and FCA solutions’ exact expected cost, UCA + Post and FCA. We observe that the bound provided by UCA converges very fast to the optimum, especially when the customer realization probability is high. Moreover, the expected cost of either algorithm’s solution is quite close to the optimum, even for $K = 1$.

Table 3 presents the geometric mean of relative gaps across instances of three solutions: the UCA solution (with exact expected cost), the FCA solution, and the expected cost of the deterministic VRPTW solution that ignores probabilities and minimizes the cost as if all customers will require
Figure 2: Convergence of UCA lower bound and solutions’ expected cost, by probability.

visits. The gaps are calculated with respect to the best possible lower bound computed by UCA for any value of $K$, where we include all instances we could solve for that $K$; this means the number of instances included in an average may vary by value of $K$. We show results for $p = 0.7$, with similar tables for $p \in \{0.5, 0.9\}$ in the Appendix, and separate results by instance type (C and R); recall that instances of type C have clustered customer locations, while type-R instances have uniformly distributed customer locations.

In all cases, the expected cost of solutions produced by UCA and FCA are within 5% of optimal on average, often much closer, and both are consistently better than the solution given by the deterministic instance. Either algorithm can produce the better solution on a particular instance; we detect no clear pattern of one producing better solutions than the other, but overall UCA has a slight advantage. Unsurprisingly, the number of customers $n$ impacts the solutions’ gap, with bigger instances having larger gaps. More interestingly, the instance type significantly affects the solutions’ quality, with gaps for type R on average much tighter than for type C. Unlike the results
Table 3: Average relative gap of solution expected cost for UCA, FCA and deterministic problem, for realization probability 0.7.

summarized in Figure 2, here we don’t always see a monotonically decreasing gap as $K$ increases, but this is a result of including different sets of instances for different values of $K$; in general, for a given instance we tend to see better solutions with larger $K$.

To further explore the benefit of optimizing with respect to expected costs, Figure 3 plots the percentage of instances where UCA and FCA obtain solutions with lower expected cost than the solution of the deterministic VRPTW. For example, when the customer realization probability is 0.5 and $K \geq 4$, both UCA and FCA produce better solutions in about 60% of the tested instances. The plots also emphasize that the realization probability affects this improvement percentage; for larger probabilities, it is harder for the algorithms to improve on the deterministic solution, presumably because this solution is already close to optimal. In particular, when the probability is 0.9, we only find a better solution roughly one quarter of the time, and this improvement occurs already with $K \geq 2$. In general, we also observe that UCA is slightly better than FCA at improving over the deterministic solution.

Finally, even when the difference in expected cost between the deterministic solution and our algorithms’ solutions is not large, the structure of the corresponding solutions changes drastically, and may have operational implications. In Figure 4, we plot the UCA and deterministic solutions
Figure 3: Percentage of instances where UCA and FCA improve on the deterministic solution.

for an instance of type R with $n = 15$ and $p = 0.5$. The expected cost of the solution to the deterministic problem is 314.19, and it is plotted on the left. The expected cost of the UCA solution with $K = 5$ is 304.27, and it is plotted on the right. The difference in expected cost is only about 3%, yet the solutions are very different. Specifically, both solutions use 5 planned routes, but without repeating any, and the UCA solution’s planned routes appear inefficient when viewed as deterministic routes; in particular, they include several crossings that the deterministic solution avoids. The results for this instance highlight the potentially counter-intuitive nature of probabilistic routing, where we may plan routes that appear inefficient, but whose expected cost is in fact lower than routes that appear cheaper.

5 Conclusions

We have proposed a new column generation and B&P framework to solve the VRP-PC, including two different algorithms, UCA and FCA. Both circumvent the difficulty of exactly pricing routes by using an approximate expected cost that under-estimates the true expected cost. UCA optimizes
Figure 4: Deterministic (left) and UCA (right, \( K = 5 \)) solutions for sample instance with \( n = 15 \) and \( p = 0.5 \).

with respect to the approximate expected cost, and thus provides an \textit{a posteriori} lower bound to the optimal expected cost, in addition to giving a solution. FCA uses exact expected costs but may halt when some routes still have small negative reduced cost. Both algorithms have approximate optimality guarantees in the form of \textit{a priori} additive gaps that depend on the precision of the expected cost approximation.

Our computational results suggest the \textit{a posteriori} gap provided by UCA is much tighter than the theoretical \textit{a priori} gap indicates. Furthermore, both gaps decrease quite rapidly as we increase the precision of the expected cost approximation in UCA or FCA; in our instances, an approximation with five steps (\( K = 5 \)) or fewer suffices to get a negligible gap. However, the problem’s difficulty is determined also by the number of customers and their realization probability; in particular, when the probabilities are large we can close the gap quickly. More generally, our results also suggest that UCA and FCA produce very good solutions, no more than 5% from optimal and usually much better. Both algorithms improve upon the solution of the deterministic instance in many cases, although the improvement is not always large in terms of relative gap. However, even in these cases the structure of the resulting solution may change significantly.

Our results motivate several questions for future research. One possibility is to incorporate cutting planes into our B&P framework with UCA or FCA, which may allow us to increase the size and/or difficulty of instances we can optimize. An interesting question relates to combining our approach with approaches that optimize other related stochastic routing models. Specifically,
in some settings it may be overly conservative to require that planned routes be feasible with certainty if some customers may not require service; an alternative along these lines is to allow routes with total demand that might exceed vehicle capacity, as long as the probability of this failure is low. This would imply a chance-constrained model similar to the one studied in [17], but with probabilistic routing costs. In general, the broad topic of column generation in probabilistic and \textit{a priori} optimization offers many challenging questions for the research community.

6  Appendix

We present three tables with additional results. Tables 4 and 5 are analogues of Table 3 for $p = 0.5$ and $p = 0.9$, respectively. Table 6 displays average running times for B&P algorithms with UCA and FCA.

| $n$ | $K$ | Type C | | | Type R | | |
|-----|-----|--------| | | | | |
|     | UCA Exact | FCA Best LB | | | UCA Exact | FCA Best LB | | |
| 15  | 2.75% | 2.67% | | | 1.79% | 2.04% | | |
| 2   | 1.43% | 2.28% | | | 1.24% | 1.95% | | |
| 3   | 1.01% | 1.15% | | | 1.23% | 1.19% | | |
| 4   | 0.92% | 0.89% | | | 1.14% | 1.14% | | |
| 5   | 0.78% | 0.82% | | | 1.28% | 1.12% | | |
|     |       | 2.91% | | |       | 2.21% | | |
| 25  | 4.15% | 4.19% | | | 3.17% | 3.51% | | |
| 2   | 4.44% | 4.99% | | | 2.66% | 3.34% | | |
| 3   | 3.83% | 4.47% | | | 2.45% | 2.78% | | |
| 4   | 3.1%  | 3.29% | | | 1.6%  | 1.7%  | | |
| 5   | 1.35% | 1.42% | | | 1.65% | 1.57% | | |
|     |       | 5.32% | | |       | 3.54% | | |
| 40  | 6.42% | 6.42% | | | 4.49% | 4.52% | | |
| 2   | 7.43% | 7.43% | | | 4.22% | 4.52% | | |
| 3   | 7.25% | 7.43% | | | 3.76% | 4.14% | | |
| 4   | 7.25% | 7.24% | | | 2.29% | 2.4%  | | |
| 5   | 5.29% | 5.53% | | | 1.79% | 1.43% | | |
|     |       | 7.43% | | |       | 4.66% | | |

Table 4: Average relative gap of solution expected cost for UCA, FCA and deterministic problem, for realization probability 0.5.
<table>
<thead>
<tr>
<th>n</th>
<th>K</th>
<th>Type C</th>
<th>Type R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>UCA Exact</td>
<td>FCA</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0.52%</td>
<td>0.53%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.31%</td>
<td>0.31%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.31%</td>
<td>0.31%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.31%</td>
<td>0.31%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.31%</td>
<td>0.31%</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0.97%</td>
<td>0.98%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.1%</td>
<td>2.1%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.1%</td>
<td>2.1%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.67%</td>
<td>1.77%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.56%</td>
<td>0.56%</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>2.46%</td>
<td>2.46%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.13%</td>
<td>3.13%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3.13%</td>
<td>3.13%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.76%</td>
<td>1.76%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2.29%</td>
<td>3.13%</td>
</tr>
</tbody>
</table>

Table 5: Average relative gap of solution expected cost for UCA, FCA and deterministic problem, for realization probability 0.9.

<table>
<thead>
<tr>
<th>n</th>
<th>K</th>
<th>Type C</th>
<th>Type R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>UCA</td>
<td>FCA</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>729</td>
<td>361</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1552</td>
<td>878</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2842</td>
<td>2018</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2803</td>
<td>2992</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4545</td>
<td>4054</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>6909</td>
<td>4736</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10622</td>
<td>8822</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>12714</td>
<td>11462</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>13715</td>
<td>13803</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>13794</td>
<td>13431</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>12474</td>
<td>8623</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>19704</td>
<td>16911</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>20190</td>
<td>20638</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>20669</td>
<td>21154</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>21600</td>
<td>20700</td>
</tr>
</tbody>
</table>

Table 6: Average running time for UCA and FCA in seconds.
References


