Paired Cooperative Reoptimization Strategy for the Vehicle Routing Problem with Stochastic Demands

Lin Zhu[b], Louis-Martin Rousseau[a,c], Walter Rei[a,d], Bo Li[b]

[a] *Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)*
[b] *College of Management and Economics, Tianjin University, Tianjin, China, 30072*
[c] *Mathematics and Industrial Engineering, Ecole Polytechnique de Montreal, Canada*
[d] *Management Science School, UQAM, Canada*

Abstract

In this paper, we develop a paired cooperative reoptimization (PCR) strategy to solve the vehicle routing problem with stochastic demands (VRPSD). The strategy can realize reoptimization policy under cooperation between a pair of vehicles, and it can be applied in the multivehicle situation. The PCR repeatedly triggers communication and partitioning to update the vehicle assignments given real-time customer demands. We present a bilevel Markov decision process to model the coordination of a pair of vehicles under the PCR strategy. We also propose a heuristic that dynamically alters the visiting sequence and the vehicle assignment given updated information. We compare our approach with a recent cooperation strategy in the literature. The results reveal that our PCR strategy performs better, with a cost saving of around 20% to 30%. Moreover, embedding communication can save an average of 1.22%, and applying our partitioning method rather than an alternative can save an average of 3.96%.

Keywords: Stochastic vehicle routing, Dynamic programming, Reoptimization, Heuristic

1. Introduction

Vehicle routing problems (VRPs) involve designing a set of minimal-cost routes to meet customer demands under a group of operational constraints [see, e.g., 1, 2, 3]. In classical VRPs, the demands are assumed to be known with certainty, and all the relevant information to compute the routes is available in advance. However, in practice, customer demands and several other aspects are often stochastic. Solving the problem deterministically by replacing the stochastic parameters with their expected values does not give good solutions [4]. This justifies the development of stochastic models that can construct solutions with regard to the observed informational flow (i.e., when and how the values associated with the stochastic parameters become known).

In this paper, we consider the VRP with stochastic demands (VRPSD) in which the demand is known only when the vehicle arrives at the customer location. It has many real-world applications, such as local-deposit delivery and collection from bank branches [5], home oil delivery [6], beer distribution, and garbage collection [7].

Email addresses: zhulin.sife@gmail.com (Lin Zhu), louis-martin.rousseau@polymtl.ca (Louis-Martin Rousseau), rei.walter@uqam.ca (Walter Rei)

In the VRPSD, a vehicle may reach a customer location without sufficient residual capacity to fulfill the demand, leading to a route *failure*, in which case a *recourse* action is necessary. Various recourse actions are possible: (i) replenishing the vehicle at the depot; (ii) scheduling a different vehicle to visit the customer where the failure occurred; or (iii) skipping the customer altogether (in this case a penalty is incurred). We consider (i), i.e., a driver performs a *replenishment* trip to the depot when a failure occurs.

Different modeling approaches have been developed to deal with the uncertain demands. These modeling approaches depend on the way both the routing and replenishment decisions are made, either *static* or *dynamic* [8]. For static approaches, stochastic programming with recourse (SPR) is often used [9]. It is a two-stage approach that minimizes the total cost of the planned routes and the expected recourse actions [e.g., 10]. Dynamic approaches, which apply a *reoptimization* policy [11, 20], use a Markov decision process (MDP) to model the real-time decisions, given the available vehicle capacity and the set of unvisited customers [e.g., 9, 12, 13, 14, 8, 15, 20].

Given the recent technological advances, reoptimization policies are now a viable strategy to decrease routing costs in the VRPSD context. However, efficiently solving the MDP models is challenging given the large numbers of actions, stages, and states involved. Therefore, most studies assume that a single vehicle is available.

Secomandi and Margot [8] observe that the existing literature on the VRPSD with reoptimization is scant, focusing on heuristic methods for the single-vehicle situation [e.g., 16, 17, 18, 13, 14, 8, 20]. To the best of our knowledge, only one study considers the multivehicle case: Goodson et al. [15] propose a roll-out algorithm, real-time information is used, and the customers are dynamically assigned to different vehicles when the demands are revealed.

The solution of the MDP model for the VRPSD in the multivehicle context is challenging. Dynamic routing and replenishment decisions are necessary, and the assignment of customers to vehicles should also be performed dynamically. We propose the use of two general concepts that have proved efficient for the VRPSD: partial reoptimization of the routes and paired-vehicle cooperation.

The partial reoptimization technique was proposed by Secomandi and Margot [8] for the single-vehicle VRPSD. It computes optimal policies locally for subsets of states, to be used for the dynamic routing and replenishment when the demand is revealed. The paired-vehicle cooperation is based on the paired locally coordinated (PLC) scheme [19]. The PLC forms pairs of vehicles and shares customers within each pair, giving a solution in which each customer is dynamically served by a vehicle or its partner.

We focus on developing a cooperation strategy, the *paired cooperative reoptimization* (PCR) strategy, for a single pair of vehicles. We can then solve the multivehicle problem by clustering the customers into groups and serving the customers in each group with a pair of vehicles, as suggested by Ak and Erera [19]. The PCR strategy is based on the partial reoptimization technique and adds *communication* between the two vehicles. Via effective communication, the customers are dynamically chosen to be served by one of the two vehicles when the updated information becomes available.

This paper's main contribution is the development of the PCR recourse strategy, which is formulated as a bilevel MDP. This strategy enables a pair of vehicles to dynamically serve a set of customers under a reoptimization policy.

We propose a heuristic that relies on both partial reoptimization and real-time communication to dynamically construct the routes performed by the pair of vehicles. We compare the performance of PCR strategy through a numerical study that shows the benefits, in terms of the total travel cost, with respect to other recourse strategies.

The remainder of this paper is organized as follows. In Section 2, we present our assumptions and discuss the general paired reoptimization problem. In Section 3, we give the definition of the PCR, and in Section 4 we discuss the bilevel MDP. In Section 5, we present our heuristic. Finally, in Section 6, we give the results of the computational study. We compare our algorithm with the PLC approach and describe two experiments that illustrate the cooperation of the PCR.

2. Problem definition

2.1. Notation and assumptions

In this paper, a single pair of vehicles cooperate to serve a set of customer demands. We use notation similar to that of Secomandi and Margot [8]. Given a complete network, let the set of nodes be $\{0, 1, \ldots, N\}$, with N a positive integer. Node 0 denotes the depot and $C = \{1, \ldots, N\}$ is the set of customers. The distances $d(i, j)$ between any two nodes i and j are known, symmetric, and satisfy the triangle inequality: $d(i, j) \leq d(i, l) + d(l, j)$, with l an additional node. Two vehicles with the same capacity Q, denoted $t1$ and $t2$, are initially located at the depot and must eventually return there. Let ξ_i, $i = 1, 2, \ldots, N$ be the discrete random variable that describes the demand of customer i. Its probability mass function is $p_i(e) = \Pr\{\xi_i = e\}$, $e = 0, 1, \ldots, E \leq Q$, and $p_i(e) = 0$, $e = E + 1, \ldots, Q$, with E a nonnegative integer. The customer demands ξ_i are independent of the vehicle routing/replenishment policy. The realization of ξ_i becomes known when the vehicle arrives at customer location i. The total depot capacity is at least $N \cdot E$, so that all the customers can be served.

We assume that each customer can be served by only one vehicle. Moreover, split deliveries are allowed, i.e., when a failure occurs, the vehicle delivers its existing load to the customer, then returns to the depot to reload, and subsequently completes the interrupted delivery.

The vehicles can communicate to dynamically modify their routes, and the locations, available capacities, and unvisited customers are visible to both of the vehicles. The information is shared under three assumptions. First, we ignore the time spent on loading and unloading and on planning (the vehicle assignments and the next customer to visit). Second, the vehicles are not permitted to have idle time. Third, the vehicles travel at the same speed. Therefore, at any given time, each vehicle's location and status (e.g., en route or replenishing) can be found by calculating the total distance traveled.

2.2. Formulation of general paired reoptimization problem

We describe the general problem with reference to the MDP formulations for the single-vehicle situation [20, 8] and the multivehicle case [15].

The paired-vehicle problem is a special case of the multivehicle problem, and it can be stated as follows. When a vehicle finishes serving its current customer, a new customer will be assigned, and the vehicle must decide whether to visit the new customer directly or via the depot. The new customer is chosen from the set of unassigned customers. The vehicles must coordinate their efforts by considering the influence of each decision on the other vehicle and on the future cost.

The decisions occur when a vehicle completes an assignment, not when it arrives at a new customer and observes the demand. The next location is always a customer location and not the depot.

We formulate the problem as an MDP with stages in the set $\Omega' = \{0, 1, 2, \ldots, K'\}$. Each stage $k \in \Omega' \setminus \{0\}$ starts as a vehicle finishes its current assignment. The two vehicles may complete their assignments simultaneously and trigger the next stage together. K' is the final stage that occurs when no customer is unassigned. Let the state space for the process be Ψ'. For each stage $k \in \{0, 1, 2, \ldots, K'\}$, we characterize the corresponding state as $x_k = (l_1, l_2, q_1, q_2, R_k(l_1, l_2))$. Here, l_1 and l_2 are the customer locations where the two vehicles completed their last assignments, q_1 and q_2 are the available capacities after those assignments, and $R_k(l_1, l_2)$ is the set of remaining customers at stage k. The initial system state is $x_0 = (0, 0, Q, Q, C)$ and the final system state is $x_{K'} = (l_1, l_2, q_1, q_2, \phi)$. For example, suppose the current state is $x_k = (l_1, l_2, q_1, q_2, R_k(l_1, l_2))$, and the vehicles will next serve customers j_1 and j_2. Suppose that vehicle $t1$ finishes serving customer j_1 and triggers the next stage $k + 1$, while vehicle $t2$ is either en route to customer j_2 or replenishing so as to meet j_2's demand. In this case, the state updates to $x_{k+1} = (j_1, l_2, q_1', q_2, R_{k+1}(j_1, l_2))$, where q_1' is the residual capacity of vehicle 1 that is $q_1' = q_1 - e$ (e being the realization of the demande for customer j_1).

Given state x_k, action (a_1^k, a_2^k) assigns the two vehicles to the next customer locations. Let $t \in \{1, 2\}$ represent the vehicles $t1$ and $t2$, and $z_k \subseteq \{1, 2\}$ be the set of vehicles that trigger stage k by completing their current assignments. Clearly, $z_k \neq \phi$, and $z_0 = \{1, 2\}$ indicates that both vehicles start to serve new assignments at the beginning. In addition, let the vehicles in $\{\{1, 2\} \setminus z_k\}$, which have not completed their assignments, continue on their planned routes. The set of actions available for state x_k, $k \in \Omega'$, is then

$$\begin{aligned} A(x_k) = \{(a_1^k, a_2^k) | a_t^k = j^{(1)} \text{ or } j^{(0)},\ \forall j \in R_{k-1}(l_1, l_2) \setminus \{a_1^{k-1}, a_2^{k-1}\} (k \geq 1) \\ \forall t \in z_k; a_t^k = a_t^{k-1},\ \forall t \in \{\{1, 2\} \setminus z_k\}; a_1^k \neq a_2^k\} \end{aligned} \tag{1}$$

Here, $j^{(1)}$ indicates a direct visit to customer j, and $j^{(0)}$ indicates a visit to customer j that is preceded by replenishment. Moreover, $j \in R_{k-1}(l_1, l_2) \setminus \{a_1^{k-1}, a_2^{k-1}\}$ $(k \geq 1)$ indicates that the next customer should be selected from those currently unassigned, so a_1^{k-1} and a_2^{k-1} should be excluded. It should be noted that when $k = 0$ we have $j \in C$. The equality $a_t^k = a_t^{k-1}$ $(k \geq 1)$ indicates that the vehicles en route, denoted $t \in \{\{1, 2\} \setminus z_k\}$, still follow their planned routes, and $a_1^k \neq a_2^k$ indicates that the two vehicles cannot be assigned to the same customer.

Action $(a_1^k, a_2^k) \in A(x_k)$ will generate a cost, denoted $g(x_k, a_1^k, a_2^k)$, associated with travel to the new destinations. If vehicle $t \in \{\{1, 2\} \setminus z_k\}$, the destination does not change, so $d(a_t^{k-1}, a_t^k) = 0$ and there is no cost. If $t \in z_k$, and the current location is l, the cost is $d(l, j)$ for $j^{(1)}$ and $d(l, 0) + d(0, j)$ for $j^{(0)}$. The transition probabilities, denoted $p_{x_k x_{k+1}}(a_1^k, a_2^k)$, are given by the demand probability distributions and the action selected. Let $J_k(x_k)$ denote the optimal cost-to-go or

value function in stage k; then the optimal action $(a_1^k, a_2^k)^*$ for a given state x_k is determined by the following Bellman equation:

$$(a_1^k, a_2^k)^* = \arg \min_{(a_1^k, a_2^k) \in A(x_k)} \{g(x_k, a_1^k, a_2^k) + \sum_{x_{k+1} \in \Psi'} p_{x_k x_{k+1}}(a_1^k, a_2^k) \cdot J_{k+1}(x_{k+1}) | x_k\} \quad \forall x_k \in \Psi'. \tag{2}$$

For reoptimization under the paired-vehicle condition, each movement of each vehicle will trigger a sharing of information for a total of K' interruptions. Usually K' equals the number of customers, N. It will be lower if the vehicles sometimes complete their assignments simultaneously, but the minimal value is $K' = \lceil \frac{N}{2} \rceil$.

For each interruption $k \in \{1, 2, \ldots, K'\}$, the calculation of $J_k(x_k)$ is more complicated than in the single-vehicle case, since the two vehicles share customers and make decisions that influence each other. To ease the computational burden, we must reduce the number of interruptions. Our PCR strategy reduces the number of interruptions and indicates how to schedule the vehicles to reoptimize the solution.

3. A paired cooperative reoptimization strategy for VRPSD

In the general scheme described in Section 2, only one customer can be assigned to a vehicle at the completion of the current assignment. In our PCR strategy, multiple customers can be assigned, and we allow the vehicles to operate independently until the next sharing of information. This reduces the number of interruptions.

The multiple customers will not necessarily be served by the assigned vehicle; some may subsequently be switched to the other vehicle. The overall process is as follows. We first divide the customers into two groups and assign each group to one of the vehicles. Each vehicle then serves its group of customers. When one completes its assignment it triggers a sharing of information. We then divide the remaining customers (currently assigned to the other vehicle) into two groups and assign each group to one of the vehicles. We repeat this procedure until all the customers have been served; the vehicles then return to the depot. This procedure is equivalent to dynamic vehicle assignment in the multivehicle case. When each new group of customers is formed, we use partial reoptimization [8] to determine the sequence of visits.

We use the term *communication* to refer to the time when a vehicle finishes its assignment and triggers a sharing of information. After each communication, the paired-vehicle problem is decomposed into a single-vehicle problem (i.e., the vehicles operate independently until the next communication), and this reduces the number of states in the problem. We now present the bilevel MDP formulation of our PCR strategy.

4. Bilevel MDP

The bilevel MDP has a higher level and a lower level. At the higher level, the state transitions occur at the communications, when the vehicle assignments are updated. At the lower level, the state transition is similar to that of Secomandi and Margot [8]. We must decide if the vehicle should visit its next customer directly or after replenishment.

In Section 4.1, we introduce the stages and states involved in the higher and lower levels. In Section 4.2, we explain the state transitions in the bilevel MDP. Section 4.2.1 explains the transitions from the higher to the lower level, and Section 4.2.2 explains the transitions from the lower to the higher level. Section 4.2.3 discusses transitions within the lower level, and Figure 1 in Section 4.2.4 summarizes the bilevel MDP. In Section 4.3, we introduce the initial state and the absorbing state and define the cost-to-go value at the final absorbing state. We apply dynamic-programming backward recursion to calculate the expected cost-to-go value at each stage. Finally, in Section 4.4, we describe two kinds of actions resulting from the hierarchical structure of the bilevel MDP.

4.1. Stage and state in bilevel MDP

4.1.1. Stage and state at higher level

At the higher level, at each stage the remaining customers are divided into two groups. Let $\Omega = \{0, 1, 2, \ldots, K^*\}$ be the set of stages at this level, where stage $k = 0$ ($k \in \Omega$) indicates the initial partitioning, and stage $k \in \{1, 2, \ldots, K^*\}$ represents the kth partitioning of the remaining customers, where K^* is the final communication. Further, assume that u_k ($k = 1, 2, \ldots, K^*$) is the set of remaining customers at the start of the kth communication, with $u_0 = C$ (at stage k=0) because initially all the customers are unvisited. Let α_k and $u_k \setminus \alpha_k$, with $\alpha_k \cap (u_k \setminus \alpha_k) = \phi$ define a partition of the set u_k into two customer sets after each communication and partitioning stage k ($\alpha_k \subseteq u_k, \alpha_k$ can equal ϕ). Let $n_1^k = \mid \alpha_k \mid$ and $n_2^k = \mid u_k \setminus \alpha_k \mid$ be the number of customers in the newly assigned customer sets. Therefore, the state for the higher-level stage k can be denoted $X_k = (\alpha_k,\ u_k \setminus \alpha_k)$, and the corresponding cost-to-go value is $V_k(\alpha_k, u_k \setminus \alpha_k)$.

4.1.2. Stage and state at lower level

At the lower level, the vehicles serve their customer sets α_k and $u_k \setminus \alpha_k$ independently. The stage $k \in \Omega$ is transferred from the higher level, and the sets of stages are denoted ω_1^k for vehicle $t1$ and ω_2^k for vehicle $t2$. The vehicles operate independently until the next communication (the start of higher-level stage $k+1$), and we formulate the problem as an MDP similar to that of Secomandi and Margot [8].

For vehicle $t1$, the lower level stage set ω_1^k is $\{n_1^k, n_1^k - 1, n_1^k - 2, \ldots, m_1^k\}$ in descending order, where $s_1^k \in \omega_1^k$ is the number of unvisited customers in the current customer set α_k. The final stage m_1^k ($0 \le m_1^k \le n_1^k$) indicates that m_1^k customers are unvisited at the start of higher-level stage $k+1$. Similarly, $\omega_2^k = \{n_2^k, n_2^k - 1, n_2^k - 2, \ldots, m_2^k\}$ ($0 \le m_2^k \le n_2^k$) is the lower level stage set for vehicle $t2$, with $s_2^k \in \omega_2^k$ the number of unvisited customers in $u_k \setminus \alpha_k$. Note that either m_1^k or m_2^k is 0, indicating that a vehicle has completed its current assignment. The new unvisited customer set is u_{k+1}, and it will be partitioned into α_{k+1} and $u_{k+1} \setminus \alpha_{k+1}$ at the higher level.

Because α_k and $u_k \setminus \alpha_k$ are served independently, the problem reduces to the single-vehicle situation discussed in [8], and we use similar definitions. The states for the lower level stages s_1^k ($s_1^k \ne n_1^k$) and s_2^k ($s_2^k \ne n_2^k$) are $x_{s_1^k} = (l_1^k, q_1^k, R_{s_1^k}(l_1^k))$ and $x_{s_2^k} = (l_2^k, q_2^k, R_{s_2^k}(l_2^k))$, where $l_1^k \in \alpha_k$ and $l_2^k \in u_k \setminus \alpha_k$ are the current customers, and q_1^k and $q_2^k \in \{0, 1, \ldots, Q\}$ are the available capacities after the demands of customers l_1^k and l_2^k have been satisfied. $R_{s_1^k}(l_1^k) \subset \alpha_k$ and $R_{s_2^k}(l_2^k) \subset u_k \setminus \alpha_k$ are the remaining customers when the vehicles are at the location of customers l_1^k and l_2^k in lower

level stages s_1^k and s_2^k. Further, $v_{s_1^k}(l_1^k, q_1^k, R_{s_1^k}(l_1^k))$ and $v_{s_2^k}(l_2^k, q_2^k, R_{s_2^k}(l_2^k))$ are the cost-to-go values at states $x_{s_1^k}$ and $x_{s_2^k}$; we will sometimes simplify this notation to $v_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))$ and $v_{s_2^k}(l_2, q_2, R_{s_2^k}(l_2))$.

Note that the initial and final states are different from those in [8]. When the higher-level stage $k = 0$, the initial states can be denoted $x_{n_1^0=|\alpha_0|} = (0, Q, \alpha_0)$ and $x_{n_2^0=|C\setminus\alpha_0|} = (0, Q, C \setminus \alpha_0)$, similarly to the definitions in [8]. But, when $k \geq 1$, the states are represented by $x_{n_1^k=|\alpha_k|} = (l_1^{k-1}, q_1^{k-1}, \alpha_k)$ and $x_{n_2^k=|u_k\setminus\alpha_k|} = (l_2^{k-1}, q_2^{k-1}, u_k \setminus \alpha_k)$. Before serving customers α_k and $u_k \setminus \alpha_k$ ($k \geq 1$), the vehicles are approaching customers l_1^{k-1} and l_2^{k-1}, which are the customers at the final states of the previous higher level stage $k-1$, customers l_1^{k-1} and l_2^{k-1} are not included in α_k and $u_k \setminus \alpha_k$ so they can be treated as exterior points, as the depot is when $k = 0$ ($0 \notin \alpha_0$ and $0 \notin C \setminus \alpha_0$). Moreover, when $k = 0$ the vehicles are initially fully loaded, but when $k \geq 1$ the initial capacity is the available capacity after the final customer at higher-level stage $k-1$ has been served; we denote these capacities q_1^{k-1} and q_2^{k-1}.

The states at the final stages, m_1^k and m_2^k ($0 \leq k \leq K^*$), are $x_{m_1^k} = (l_1^k, q_1^k, R_{m_1^k}(l_1^k))$ and $x_{m_2^k} = (l_2^k, q_2^k, R_{m_2^k}(l_2^k))$. Both $R_{m_1^k}(l_1^k)$ and $R_{m_2^k}(l_2^k)$ will be ϕ only if the vehicles finish their assignments simultaneously. Moreover, the final action of the last stages m_1^k and m_2^k may not be a return to the depot if there are unvisited customers.

4.2. State transition in bilevel MDP

We now consider transitions between the levels of the MDP and within the lower level. As the higher level moves from stage k to $k+1$, $k = 0, 1, \ldots, K^* - 1$, the cost-to-go value changes from $V_k(\alpha_k, u_k \setminus \alpha_k)$ to $V_{k+1}(\alpha_{k+1}, u_{k+1} \setminus \alpha_{k+1})$. As the lower level moves from stage s_1^k to $s_1^k - 1$ and from stage s_2^k to $s_2^k - 1$, the cost-to-go values change from $v_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))$ to $v_{s_1^k-1}(j_1, q_1', R_{s_1^k-1}(j_1; l_1))$ and from $v_{s_2^k}(l_2, q_2, R_{s_2^k}(l_2))$ to $v_{s_2^k-1}(j_2, q_2', R_{s_2^k-1}(j_2; l_2))$, where $j_1 \in R_{s_1^k}(l_1)$, $j_2 \in R_{s_2^k}(l_2)$, $R_{s_1^k-1}(j_1; l_1) = R_{s_1^k}(l_1) \setminus \{j_1\}$, and $R_{s_2^k-1}(j_2; l_2) = R_{s_2^k}(l_2) \setminus \{j_2\}$.

4.2.1. Transition from higher level to lower level

This transition occurs after u_k has been divided into two sets α_k and $u_k \setminus \alpha_k$, and the vehicle assignments have been updated accordingly. We transition from higher level state $X_k = (\alpha_k, u_k \setminus \alpha_k)$ to lower level states $x_{n_1^k} = (l_1^{k-1}, q_1^{k-1}, \alpha_k)$ and $x_{n_2^k} = (l_2^{k-1}, q_2^{k-1}, u_k \setminus \alpha_k)$, ($n_1^k =\mid \alpha_k \mid$, $n_2^k =\mid u_k \setminus \alpha_k \mid$). This transition does not involve any vehicle movements, so there is no associated cost. The higher level state $X_k = (\alpha_k, u_k \setminus \alpha_k)$ has corresponding lower level states $x_{n_1^k} = (l_1^{k-1}, q_1^{k-1}, \alpha_k)$ and $x_{n_2^k} = (l_2^{k-1}, q_2^{k-1}, u_k \setminus \alpha_k)$ with

$$V_k(\alpha_k, u_k \setminus \alpha_k) = v_{n_1^k}(l_1, q_1, \alpha_k) + v_{n_2^k}(l_2, q_2, u_k \setminus \alpha_k). \tag{3}$$

4.2.2. Transition from lower level to higher level

At the end of a lower level stage a communication is triggered. We transition from higher level state $X_k = (\alpha_k, u_k \setminus \alpha_k)$ to $X_{k+1} = (\alpha_{k+1}, u_{k+1} \setminus \alpha_{k+1})$.

Assume that at stage k ($0 \leq k \leq K^* - 1$), vehicle $t1$ finishes its task α_k and triggers the communication, and the final stages are m_1^k and m_2^k. Vehicle $t1$ is located at its final customer, and its state is $x_{m_1^k=0} = (l_1, q_1, \phi)$ ($l_1 \in \alpha_k$). However, vehicle $t2$ may be at customer i_2 ($i_2 \in u_k \setminus \alpha_k$) if $\xi_{i_2} \leq q_2$, on a replenishment trip if $\xi_{i_2} > q_2$, or en route to the next

customer j_2 ($j_2 \in u_k \setminus \alpha_k$, $j_2 \neq i_2$). If $t2$ is serving customer i_2, the final state is $x_{\bar{s}_2^k} = (i_2, q_2, R_{\bar{s}_2^k}(i_2))$ ($n_2^k \geq \bar{s}_2^k \geq 0$). If it is en route to customer j_2, the final state is $x_{\bar{s}_2^k-1} = (j_2, q_2', R_{\bar{s}_2^k-1}(j_2))$=$(j_2, q_2', R_{\bar{s}_2^k-1}(j_2; i_2))$ =$(j_2, q_2', R_{\bar{s}_2^k}(i_2) \setminus \{j_2\})$, which indicates that it should complete the service of j_2 before updating its assignment.

Thus, the final states are as follows. If $t2$ is serving i_2, $(x_{m_1^k=0}, x_{\bar{s}_2^k})$=$((l_1, q_1, \phi), (i_2, q_2, R_{\bar{s}_2^k}(i_2)))$ with $m_2^k = \bar{s}_2^k$ and $u_{k+1} = \phi \cup R_{\bar{s}_2^k}(i_2) = R_{\bar{s}_2^k}(i_2)$. There is no cost associated with the partitioning of u_{k+1} into $\alpha_{k+1} \cup (u_{k+1} \setminus \alpha_{k+1})$, so the cost-to-go value is

$$v_{m_1^k=0}(l_1, q_1, \phi) + v_{m_2^k=\bar{s}_2^k}(i_2, q_2, R_{\bar{s}_2^k}(i_2)) = V_{k+1}(\alpha_{k+1}, u_{k+1} \setminus \alpha_{k+1}). \tag{4}$$

If $t2$ is en route to j_2, $(x_{m_1^k=0}, x_{\bar{s}_2^k-1})$=$((l_1, q_1, \phi), (j_2, q_2', R_{\bar{s}_2^k-1}(j_2; i_2)))$ with $m_2^k = \bar{s}_2^k - 1$, $u_{k+1} = \phi \cup R_{\bar{s}_2^k-1}(j_2; i_2) = R_{\bar{s}_2^k-1}(j_2; i_2) = R_{\bar{s}_2^k}(i_2) \setminus \{j_2\}$. The cost-to-go value is

$$v_{m_1^k=0}(l_1, q_1, \phi) + v_{m_2^k=\bar{s}_2^k-1}(j_2, q_2', R_{\bar{s}_2^k-1}(j_2; i_2)) = V_{k+1}(\alpha_{k+1}, u_{k+1} \setminus \alpha_{k+1}). \tag{5}$$

4.2.3. Transition within lower level

At the lower level there are transitions for $x_{s_1^k} = (l_1, q_1, R_{s_1^k}(l_1))$, from the initial stage $s_1^k = n_1^k$ to the final stage $s_1^k = m_1^k$ and $x_{s_2^k} = (l_2, q_2, R_{s_2^k}(l_2))$, from the initial stage $s_2^k = n_2^k$ to the final stage $s_2^k = m_2^k$. The transitions for the two vehicles can be treated separately, following the single-vehicle situation in [8]. For example, for vehicle $t1$, the cost-to-go value depends on whether the next customer is visited directly or after replenishment. The optimal state transition should satisfy the Bellman equations below.

For each stage, $n_1^k - 1 \geq s_1^k \geq m_1^k + 1$, the optimal cost-to-go function for each state $x_{s_1^k} = (l_1, q_1, R_{s_1^k}(l_1))$ is

$$v_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)) = \min\{v_{s_1^k}^{\mathcal{D}}(l_1, q_1, R_{s_1^k}(l_1)),\ v_{s_1^k}^{\mathcal{R}}(l_1, q_1, R_{s_1^k}(l_1))\} \tag{6}$$

where $v_{s_1^k}^{\mathcal{D}}(l_1, q_1, R_{s_1^k}(l_1))$ and $v_{s_1^k}^{\mathcal{R}}(l_1, q_1, R_{s_1^k}(l_1))$ are the cost-to-go values at stage s_1^k, corresponding to visiting the next customer directly or after replenishment:

$$\begin{aligned} v_{s_1^k}^{\mathcal{D}}(l_1, q_1, R_{s_1^k}(l_1)) = \min_{j \in R_{s_1^k}(l_1)} \{ & d(l_1, j) + \sum_{e=0}^{q_1} p_j(e) v_{s_1^k-1}(j, q_1 - e, R_{s_1^k-1}(j; l_1)) + \\ & \sum_{e=q_1+1}^{E} p_j(e)[v_{s_1^k-1}(j, q_1 + Q - e, R_{s_1^k-1}(j; l_1)) + 2d(j, 0)]\}, \\ v_{s_1^k}^{\mathcal{R}}(l_1, q_1, R_{s_1^k}(l_1)) = \min_{j \in R_{s_1^k}(l_1)} \{ & d(l_1, 0) + d(0, j) + \sum_{e=0}^{E} p_j(e) v_{s_1^k-1}(j, Q - e, R_{s_1^k-1}(j; l_1))\} \end{aligned} \tag{7}$$

For the final state $x_{m_1^k} = (l_1^k, q_1^k, R_{m_1^k}(l_1^k))$ and the initial state $x_{n_1^k} = (l_1^{k-1}, q_1^{k-1}, \alpha_k)$, the rule differs from that in [8]. For the final state, the cost-to-go value $v_{m_1^k}(l_1, q_1, R_{m_1^k}(l_1))$ is determined by (4) or (5) depending on the status of the vehicle. For the initial state, since the vehicle departs from customer l_1^{k-1} ($l_1^{k-1} \notin \alpha_k$) not the depot 0, the function $v_{n_1^k}(l_1^{k-1}, q_1^{k-1}, \alpha_k)$ is found from (6) and (7), since the vehicle may visit the next customer directly $v_{n_1^k}^{\mathcal{D}}(l_1^{k-1}, q_1^{k-1}, \alpha_k)$ or after replenishment $v_{n_1^k}^{\mathcal{R}}(l_1^{k-1}, q_1^{k-1}, \alpha_k)$.

4.2.4. Overall bilevel MDP

The overall process, from higher level stage $k\,(0 \le k < K^* - 1)$ to lower level stages n_1^k to m_1^k and n_2^k to m_2^k, and on to higher level stage $k+1$ is illustrated in Figure 1.

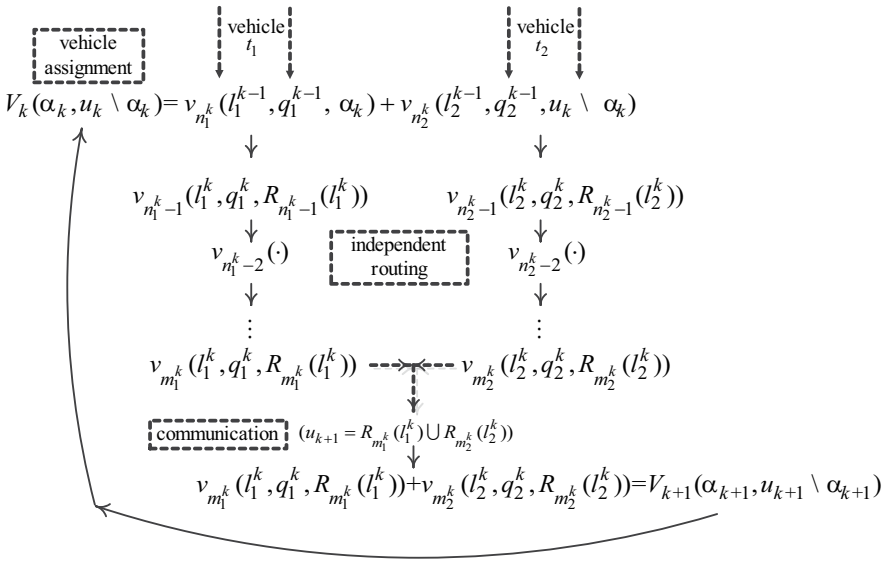


Fig. 1 State transitions of bilevel MDP

The transition from higher level stage k to $k+1$ is carried out as shown in the figure, and then the process is repeated for stage $k+1$, and so on to the final stage $k = K^*$.

4.3. Initial and final states

At the initial state the vehicles leave the depot to carry out their first tasks α_0 and $C \setminus \alpha_0$. The cost-to-go function is similar to that in (3):

$$V_{k=0}(\alpha_0, C \setminus \alpha_0) = v_{n_1^0=|\alpha_0|}(0, Q, \alpha_0) + v_{n_2^0=|C\setminus\alpha_0|}(0, Q, C \setminus \alpha_0). \tag{8}$$

At the final state the final communication K^* is triggered, and for this communication $R_{m_1^{K^*-1}=0}(l_1) = R_{m_2^{K^*-1}=0}(l_2) = \phi$ indicating that all the customers have been visited. The vehicles then return to the depot. The cost-to-go value is

$$\begin{aligned} &v_{m_1^{K^*-1}=0}(l_1, q_1, \phi) + v_{m_2^{K^*-1}=0}(l_2, q_2, \phi) = V_{K^*}(\phi, \phi) \\ &\text{where} \quad \begin{cases} v_{m_1^{K^*-1}=0}(l_1, q_1, \phi) = d(l_1, 0), \\ v_{m_2^{K^*-1}=0}(l_2, q_2, \phi) = d(l_2, 0), \end{cases} \quad \forall\, l_1 \in \alpha_{K^*-1}, l_2 \in u_{K^*-1} \setminus \alpha_{K^*-1}; \forall\, q_1, q_2 \le Q. \end{aligned} \tag{9}$$

The bilevel MDP can be solved by dynamic-programming backward recursion, and at each stage the expected cost-to-go values can all be calculated. The optimal policy under the PCR strategy is now determined by defining the actions that can occur when transitioning from state to state for each stage and each level. This is the subject of the following subsection.

4.4. Action definitions

There are two types of actions. The first is the lower level decision (see [8]) to replenish or to visit the next customer directly. The second occurs when a communication is triggered and the remaining customers must be divided into two groups.

For the first action, we take vehicle $t1$ as an example. For state $x_{s_1^k} = (l_1, q_1, R_{s_1^k}(l_1))$, the optimal action is

$$a_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)) = \begin{cases} j^{\mathcal{D}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1), \mathcal{D}) & \text{if } v^{\mathcal{D}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)) \le v^{\mathcal{R}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)), \\ j^{\mathcal{R}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1), \mathcal{R}) & \text{if } v^{\mathcal{D}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)) > v^{\mathcal{R}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)). \end{cases} \tag{10}$$

where

$$\begin{aligned} j^{\mathcal{D}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1), \mathcal{D}) = \arg \min_{j \in R_{s_1^k}(l_1)} \{ & d(l_1, j) + \sum_{e=0}^{q_1} p_j(e) v_{s_1^k - 1}(j, q_1 - e, R_{s_1^k - 1}(j; l_1)) + \\ & \sum_{e=q_1+1}^{E} p_j(e) [v_{s_1^k - 1}(j, q_1 + Q - e, R_{s_1^k - 1}(j; l_1)) + 2d(j, 0)] \}, \end{aligned}$$

$$j^{\mathcal{R}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1), \mathcal{R}) = \arg \min_{j \in R_{s_1^k}(l_1)} \{ d(l_1, 0) + d(0, j) + \sum_{e=0}^{E} p_j(e) v_{s_1^k - 1}(j, Q - e, R_{s_1^k - 1}(j; l_1)) \} \tag{11}$$

Note that $j^{\mathcal{D}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1), \mathcal{D})$ and $j^{\mathcal{R}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1), \mathcal{R})$ denote the optimal customer to visit next for cases $\mathcal{D}$ and $\mathcal{R}$.

The optimal action at stage K^* is obvious: the vehicles must return to the depot (see (9)). The optimal actions for the two vehicles at their initial states, $x_{n_1^0} = (0, Q, \alpha_0)$ and $x_{n_2^0} = (0, Q, C \setminus \alpha_0)$, are

$$\begin{aligned} j_{n_1^0}(0, Q, \alpha_0) &= \arg \min_{j \in \alpha_0} \{ d(0, j) + \sum_{e=0}^{E} p_j(e) v_{n_1^0 - 1}(j, Q - e, \alpha_0 \setminus \{j\}) \}, \\ j_{n_2^0}(0, Q, C \setminus \alpha_0) &= \arg \min_{j \in C \setminus \alpha_0} \{ d(0, j) + \sum_{e=0}^{E} p_j(e) v_{n_2^0 - 1}(j, Q - e, (C \setminus \alpha_0) \setminus \{j\}) \}. \end{aligned} \tag{12}$$

For the second action, we determine the new customer groups by minimizing the total cost-to-go value. For example, for the kth partitioning, the optimal action is

$$A_k(\alpha_k, u_k \setminus \alpha_k) = \arg \min_{\alpha_k \subseteq u_k} V_k(\alpha_k, u_k \setminus \alpha_k) = \arg \min_{\alpha_k \subseteq u_k} \{ v_{|\alpha_k|}(l_1^{k-1}, q_1^{k-1}, \alpha_k) + v_{|u_k \setminus \alpha_k|}(l_2^{k-1}, q_2^{k-1}, u_k \setminus \alpha_k) \} \quad (k \ge 1) \tag{13}$$

where $v_{|\alpha_k|}(l_1^{k-1}, q_1^{k-1}, \alpha_k)$ and $v_{|u_k \setminus \alpha_k|}(l_2^{k-1}, q_2^{k-1}, u_k \setminus \alpha_k)$ represent the cost-to-go values when the customer sets α_k and $u_k \setminus \alpha_k$ are served independently. Note that α_k can be ϕ; if only one customer j remains, i.e., $u_k = \{j\}$, the partitioning will be $\alpha_k = \phi$, $u_k \setminus \alpha_k = \{j\}$ or $\alpha_k = \{j\}$, $u_k \setminus \alpha_k = \phi$. For the initial state, the action is

$$A_0(\alpha_0, C \setminus \alpha_0) = \arg \min_{\alpha_0 \subseteq C} V_0(\alpha_0, C \setminus \alpha_0) = \arg \min_{\alpha_0 \subseteq C} \{ v_{|\alpha_0|}(0, Q, \alpha_0) + v_{|C \setminus \alpha_0|}(0, Q, C \setminus \alpha_0) \}. \tag{14}$$

5. Heuristic algorithm for PCR strategy

We now propose a heuristic algorithm for the PCR strategy. The procedure relies on approximation techniques that compute the cost-to-go values at the different stages of the MDP and on a dynamic scheduling approach that divides the remaining customers between the vehicles, given the revealed demands and expected traveled distances.

5.1. Approximate partial reoptimization

We use approximation techniques to measure the cost-to-go values because the state space is extremely large. Take vehicle $t1$ as an example. Figure 2 illustrates the transitions from any state in the kth lower-level MDP to the absorbing state $v_0(l_1^{K^*-1}, q_1^{K^*-1}, \phi)$ and shows how the approximation is calculated.

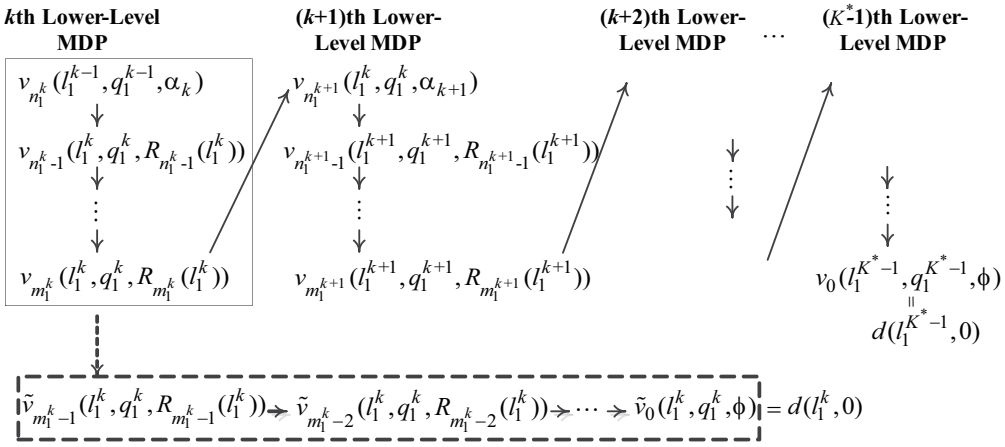


Fig. 2 Approximation of cost-to-go value in bilevel MDP

The approximate cost-to-go values are denoted $\tilde{v}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))$ and $\tilde{v}_{s_2^k}(l_2, q_2, R_{s_2^k}(l_2))$, $n_1^k \geq s_1^k \geq m_1^k$, $n_2^k \geq s_2^k \geq m_2^k$, $0 \leq k \leq K^* - 1$. They replace the exact values $v_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))$ and $v_{s_2^k}(l_2, q_2, R_{s_2^k}(l_2))$ in the calculations of the optimal actions. As Fig. 2 shows, when the kth communication is triggered, u_k is observed and α_k is immediately assigned to vehicle $t1$. Although m_1^k customers will remain at the next communication $k+1$, we assume that the customers in α_k are all served. Let the cost-to-go value when all the customers in α_k have been served be $d(l_1^k, 0)$ (l_1^k is the last customer in α_k to be visited). Then

$$\tilde{v}_0(l_1^k, q_1^k, \phi) = d(l_1^k, 0), \quad \forall l_1^k \in \alpha_k, \forall q_1^k \leq Q. \tag{15}$$

By backward recursion, we can find $\tilde{v}_1(l_1^k, q_1^k, R_1(l_1^k)), \ldots, \tilde{v}_{m_1^k-1}(l_1^k, q_1^k, R_{m_1^k-1}(l_1^k))$ and thence $\tilde{v}_{m_1^k}(l_1^k, q_1^k, R_{m_1^k}(l_1^k))$, $\tilde{v}_{m_1^k+1}(l_1^k, q_1^k, R_{m_1^k+1}(l_1^k)), \ldots$, until we have the initial cost-to-go value $\tilde{v}_{n_1^k}(l_1^{k-1}, q_1^{k-1}, \alpha_k)$. Then, using (6) and (7), $\tilde{v}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))$, $0 < s_1^k \leq n_1^k$ is given by

$$\tilde{v}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)) = \min\{\tilde{v}^{\mathcal{D}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)),\ \tilde{v}^{\mathcal{R}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))\}, \tag{16}$$

$$\begin{aligned}\tilde{v}^{\mathcal{D}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)) = \min_{j \in R_{s_1^k}(l_1)} \{d(l_1, j) + \sum_{e=0}^{q_1} p_j(e)\tilde{v}_{s_1^k-1}(j, q_1 - e, R_{s_1^k-1}(j; l_1)) + \\ \sum_{e=q_1+1}^{E} p_j(e)[\tilde{v}_{s_1^k-1}(j, q_1 + Q - e, R_{s_1^k-1}(j; l_1)) + 2d(j, 0)]\},\end{aligned}$$

$$\tilde{v}^{\mathcal{R}}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1)) = \min_{j \in R_{s_1^k}(l_1)} \{d(l_1, 0) + d(0, j) + \sum_{e=0}^{E} p_j(e)\tilde{v}_{s_1^k-1}(j, Q - e, R_{s_1^k-1}(j; l_1))\} \tag{17}$$

$\tilde{v}^{\mathcal{D}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1))$ and $\tilde{v}^{\mathcal{R}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1))$ are the approximate values of $v^{\mathcal{D}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1))$ and $v^{\mathcal{R}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1))$.

Our approximate partial reoptimization method is designed to estimate the cost-to-go value at each stage and state for each vehicle. When α_k and $u_k \setminus \alpha_k$ $(0 \leq k \leq K^* - 1)$ are assigned to the vehicles, the approximate partial reoptimization is applied to calculate the cost-to-go values $\tilde{v}_{n^k_1}(l^{k-1}_1, q^{k-1}_1, \alpha_k)$, $\tilde{v}_{n^k_1-1}(l^k_1, q^k_1, R_{n^k_1-1}(l^k_1))$, …, $\tilde{v}_0(l^k_1, q^k_1, \phi)$, and $\tilde{v}_{n^k_2}(l^{k-1}_2, q^{k-1}_2, u_k \setminus \alpha_k)$, $\tilde{v}_{n^k_2-1}(l^k_2, q^k_2, R_{n^k_2-1}(l^k_2)), \ldots, \tilde{v}_0(l^k_2, q^k_2, \phi)$. These cost-to-go values are then used to decide the optimal actions at each stage s^k_1 and s^k_2 given the revealed state. For example, for vehicle $t1$ $(\forall n^k_1 \geq s^k_1 \geq 0)$, the action is

$$a_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1)) = \begin{cases} sf^{\mathcal{D}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1), \mathcal{D}) & \text{if } \tilde{v}^{\mathcal{D}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1)) \leq \tilde{v}^{\mathcal{R}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1)), \\ f^{\mathcal{R}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1), \mathcal{R}) & \text{if } \tilde{v}^{\mathcal{D}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1)) > \tilde{v}^{\mathcal{R}}_{s^k_1}(l_1, q_1, R_{s^k_1}(l_1)). \end{cases} \tag{18}$$

When one of the vehicles reaches the last stage of the kth lower level, either s^k_1 or s^k_2 is 0, we can find the new customer set u_{k+1}, and form α_{k+1} and $u_{k+1} \setminus \alpha_{k+1}$. Then, the new round of calculating the approximate cost-to-go values $\tilde{v}_{n^{k+1}_1=|\alpha_{k+1}|}(l^k_1, q^k_1, \alpha_{k+1})$, $\tilde{v}_{n^{k+1}_1-1}(l^{k+1}_1, q^{k+1}_1, R_{n^{k+1}_1-1}(l^{k+1}_1)), \ldots, \tilde{v}_0(l^{k+1}_1, q^{k+1}_1, \phi)$ and $\tilde{v}_{n^{k+1}_2=|u_{k+1}\setminus\alpha_{k+1}|}(l^k_2, q^k_2, u_{k+1} \setminus \alpha_{k+1})$,$\tilde{v}_{n^{k+1}_2-1}(l^{k+1}_2, q^{k+1}_2, R_{n^{k+1}_2-1}(l^{k+1}_2)), \ldots, \tilde{v}_0(l^{k+1}_2, q^{k+1}_2, \phi)$ are executed. Upon the updated values, we proceed as before. This recalculation and replacement of the existing cost-to-go values to guide vehicle decisions repeats, until there is no customer remaining unserved, the vehicles finally return to the depot.

5.2. Solving the lower-level MDP

As previously presented in Section 3, whenever communication occurs between the vehicles, the unserved customer set (i.e., u_k for $k \in \{1, 2, \ldots, K^*\}$) is partitioned in two subsets (i.e., α_k and $u_k \setminus \alpha_k$) that are then assigned to the vehicles. Each vehicle then proceeds to serve its associated subset under the reoptimization strategy. To solve these VRPSD, we apply the partitioning heuristic (PH), developed in [8], which was shown to be an efficient solution approach for the problem. In this subsection, we briefly recall the general principles behind this algorithm.

The PH heuristic produces an optimal partial reoptimization policy by applying backward dynamic programming on a restricted sate set for the VRPSD. This restricted state set is obtained by first producing a complete tour of the customers to serve. Using the obtained tour, a set of disjoint contiguous blocks of customers is generated. The state set is then produced by considering that the sequence of customer blocks in the tour remains unchanged, but that the order of the customer visits within each block is dynamically reoptimized. When applying the PH heuristic, the block size (i.e., parameter M) is assumed fixed beforehand. As originally noted in [8], if parameter M is set to one, then applying PH is equivalent to evaluating the tour by exclusively considering the restocking option. On the other hand, if parameter M is set to the total number of customers to serve (i.e., n^k_1 or n^k_2 for either the first or second vehicle), then PH produces an optimal reoptimization policy. In the present context, parameter M is fixed to six, which was found to be an efficient value to apply the heuristic. By using this value, PH is able to obtain good quality results, while keeping the computational effort needed to solve the problems under control.

5.3. Approximate partition

The higher level transitions divide the remaining customers between the vehicles. The cost-to-go values should be evaluated based on all possible partitions $X_k = (\alpha_k, u_k \backslash \alpha_k)\ (\forall \alpha_k \subseteq u_k, k \in \{0, 1, \ldots, K^* - 1\})$, but this is computationally prohibitive. Therefore, we use an approximate approach.

Our partition is constructed based on the a priori route that traverses all the customers of each $u_k\ (0 \leq k \leq K^* - 1)$. The expected distance of a fixed route under the classical detour-to-depot policy (e.g., [21]) is applied to approximate the cost-to-go value in (13) and (14). The expected distance is the fixed route distance plus the total length of the expected recourse actions. Let $r^k = \{l, i_1^k, i_2^k, \ldots, i_{j^*}^k, i_{j^*+1}^k, \ldots, i_n^k, l'\}$ be the sequence of the a priori route, which starts at l and ends at l' $(l, l' \notin u_k)$, and let $\{i_1^k, i_2^k, \ldots, i_{j^*}^k, i_{j^*+1}^k, \ldots, i_n^k\}$ be a permutation of the customers in u_k. Let $dis^0(l, i^k, j^k)$ and $dis^1(l, i^k, j^k)$ respectively represent the total expected distances when the permutation is traversed in the given order or in the opposite direction, where l is the starting point, i^k is the first customer, and j^k is the last customer. We use the rollout algorithm (RA) in [22] to generate a good a priori route r^k for each customer set $u_k\ (0 \leq k \leq K^* - 1)$. Our approximate partitioning algorithm is as follows:

Step 1: Use RA to generate the a priori route r^k for customer set u_k;
this gives the sequence $r^k = \{l, i_1^k, i_2^k, \ldots, i_{j^*}^k, i_{j^*+1}^k, \ldots, i_n^k, l'\}$.

Step 2: Partition u_k into α_k and $u_k \setminus \alpha_k$ by finding a customer $i_{j^*}^k$ $(0 \leq j^* \leq n)$, that minimizes the gap $| dis^0(l, i_1^k, i_{j^*}^k) - dis^1(l', i_n^k, i_{j^*+1}^k) |$, leading to $\alpha_k = \{i_1^k, i_2^k, \ldots, i_{j^*}^k\}$ and $u_k \setminus \alpha_k = \{i_n^k, i_{n-1}^k, \ldots, i_{j^*+2}^k, i_{j^*+1}^k\}$.

We use $(\arg \min_{0 \leq j^* \leq n} \{| dis^0(l, i_1^k, i_{j^*}^k) - dis^1(l', i_n^k, i_{j^*+1}^k) |\})$ to approximate (13) and (14). Note that in Step 1 the starting and ending points depend on the partition. We have $r^0 = \{0, i_1^0, i_2^0, \ldots, i_{j^*}^0, i_{j^*+1}^0, \ldots, i_n^0, 0\}$ for the initial partition where $u_0 = C$ ($l = l' = 0$ indicates that the vehicle starts and ends at the depot), and $r^k = \{l_1^{k-1}, i_1^k, i_2^k, \ldots, i_{j^*}^k, i_{j^*+1}^k, \ldots, i_n^k, l_2^{k-1}\}$ for the kth partition, $1 \leq k \leq K^* - 1$ (l_1^{k-1} and l_2^{k-1} are the last customers visited before the assignments are updated). In Step 2, we assume that the two vehicles travel in opposite directions. Therefore, $dis^0(l, i_1^k, i_{j^*}^k)$ indicates that vehicle $t1$ follows the sequence of the a priori route, traveling from the previous location l to i_1^k and continuing to $i_{j^*}^k$; and $dis^1(l', i_n^k, i_{j^*+1}^k)$ is the expected length of the sequence $\{l', i_n^k, i_{n-1}^k, \ldots, i_{j^*+2}^k, i_{j^*+1}^k\}$. Here $j^* = 0$ and $j^* = n$ correspond to the partitions (ϕ, u_k) and (u_k, ϕ) respectively. We have $dis^0(l, i_1^k, i_{j^*=0}^k)$ equal to $d(l, 0)$, the distance to the depot, when $\alpha_k = \phi$, and $dis^1(l', i_n^k, i_{j^*+1=n+1}^k)$ equal to $d(l', 0)$ when $u_k \setminus \alpha_k = \phi$. Finally, the subroutes $r_1^k = \{l_1^{k-1}, i_1^k, i_2^k, \ldots, i_{j^*}^k\}$ (or $r_1^0 = \{0, i_1^0, i_2^0, \ldots, i_{j^*}^0\}$) and $r_2^k = \{l_2^{k-1}, i_n^k, i_{n-1}^k, \ldots, i_{j^*+2}^k, i_{j^*+1}^k\}$ (or $r_2^0 = \{0, i_n^0, i_{n-1}^0, \ldots, i_{j^*+2}^0, i_{j^*+1}^0\}$) are the a priori routes of vehicles $t1$ and $t2$ for α_k and $u_k \setminus \alpha_k$ respectively. We use these in the approximate partial reoptimization to calculate the cost-to-go values and the customers served.

5.4. Heuristic design for PCR strategy

We now describe the heuristic for the PCR strategy. Assume that $begin_1^k$ and $begin_2^k$ represent the starting locations of $t1$ and $t2$ at each higher level stage k ($k \in \{0, 1, \ldots, K^* - 1\}$). These locations correspond to l_1^{k-1} and l_2^{k-1} in the initial states $x_{n_1^k = |\alpha_k|} = (l_1^{k-1}, q_1^{k-1}, \alpha_k)$ and $x_{n_2^k = |\ u_k \setminus \alpha_k|} = (l_2^{k-1}, q_2^{k-1}, u_k \setminus \alpha_k)$, and in particular for $k = 0$ we have $begin_1^0 = 0$ and

$begin_2^0 = 0$, indicating departures from the depot. Let $total\_len_{t\in\{1,2\}}$ be the cumulative distances traveled by vehicles $t1$ and $t2$, and let $\Delta len_{t\in\{1,2\}}(r)$ indicate the distance that the corresponding vehicle will travel on route r based on the PCR strategy. Moreover, let $subroute(r, i, j)$ represent the subroute of route r that covers customer i to customer j.

Let $r\_real_1^k$ and $r\_real_2^k$ $(k \geq 1)$ denote the actual routes of the vehicles, starting from l_1^{k-1} and l_2^{k-1} and visiting all the customers in α_k and $u_k \setminus \alpha_k$. Note that $r\_real_1^k$ $(r\_real_2^k)$ is the actual route determined by the reoptimization policy, whereas r_1^k (r_2^k) is the a priori route that must be further optimized when the demands are revealed. When we write $r\_real_{t\in\{1,2\}}^k$ as an argument of $\Delta len(\cdot)$ and $subroute(\cdot, \cdot, \cdot)$, we are referring to the lengths of $r\_real_{t\in\{1,2\}}^k$ and its associated subroute. In addition, let $l_{t\in\{1,2\}}^{k(Final)}$ be the final customer of route $r\_real_{t\in\{1,2\}}^k$. The process can be stated as follows:

Step 1: Initialization

Set $u_0 = C$ and $k = 0$.

Set the starting positions $begin_1^0 = 0$ and $begin_2^0 = 0$.

Set the initial cumulative distances traveled to $total\_len_1 = 0$ and $total\_len_2 = 0$.

Step 2: Apply the approximate partitioning algorithm to divide u_k and generate the a priori routes r_1^k and r_2^k.

We use $\min_{0 \le j^* \le n} \{| \, dis^0(l, i_1^k, i_{j^*}^k) - dis^1(l', i_n^k, i_{j^*+1}^k)|\}$ to partition u_k into α_k and $u_k \setminus \alpha_k$.

The a priori routes are $r_1^k = \{l_1^{k-1}, i_1^k, i_2^k, \ldots, i_{j^*}^k\}$ (or $r_1^0 = \{0, i_1^0, i_2^0, \ldots, i_{j^*}^0\}$) and $r_2^k = \{l_2^{k-1}, i_n^k, i_{n-1}^k, \ldots, i_{j^*+2}^k, i_{j^*+1}^k\}$ (or $r_2^0 = \{0, i_n^0, i_{n-1}^0, \ldots, i_{j^*+2}^0, i_{j^*+1}^0\}$) for α_k and $u_k \setminus \alpha_k$ ($k \ge 0$), respectively.

if $r_1^k \neq \phi$ & $r_2^k \neq \phi$ (* $\alpha_k \neq \phi$ & $u_k \setminus \alpha_k \neq \phi$ *), go to Step 3.

else (* the case (ϕ, u_k) or (u_k, ϕ) *)

Let t'' be the vehicle with the empty demand ϕ and t^{**} be the vehicle with the new assignment u_k. Go to Step 5.

Step 3: Apply the approximate partial reoptimization for each vehicle.

Step 3.1: Calculate the approximate cost-to-go values $\tilde{v}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))$ and $\tilde{v}_{s_2^k}(l_2, q_2, R_{s_2^k}(l_2))$ by backward recursion, at each state and for each stage $s_1^k \in \{| \, r_1^k \, |, | \, r_1^k - 1 \, |, \ldots, 1, 0\}$ and $s_2^k \in \{| \, r_2^k \, |, | \, r_2^k - 1 \, | \, , \ldots, 1, 0\}$.

Step 3.2: Use $\tilde{v}_{s_1^k}(l_1, q_1, R_{s_1^k}(l_1))$ and $\tilde{v}_{s_2^k}(l_2, q_2, R_{s_2^k}(l_2))$ to guide the reoptimizations for $t1$ and $t2$.

Find the actual routes $r\_real_1^k$ and $r\_real_2^k$ for α_k and $u_k \setminus \alpha_k$, and the corresponding final customers in these routes, $l_1^{k(Final)} \in \alpha_k$ and $l_2^{k(Final)} \in u_k \setminus \alpha_k$.

Step 4: Determine the $(k+1)$th communication and update the customer set.

Compare $total\_len_1 + \Delta len_1(r\_real_1^k)$ and $total\_len_2 + \Delta len_2(r\_real_2^k)$; the shorter length identifies the vehicle that triggers the $(k+1)$th communication. Let $t^* = \arg\min_{t \in \{1,2\}} \{total\_len_t + \Delta len_t(r\_real_t^k)\}$ and $t' = \arg\max_{t \in \{1,2\}} \{total\_len_t + \Delta len_t(r\_real_t^k)\}$.

if $t^* = t'$

Set $u_{k+1} = \phi$, $total\_len_1 = total\_len_1 + \Delta len_1(r\_real_1^k) + d(l_1^{k(Final)}, 0)$, and $total\_len_2 = total\_len_2 + \Delta len_2(r\_real_2^k) + d(l_2^{k(Final)}, 0)$.

else

(* vehicle t^* performs the full route $r\_real_{t^*}^k$ *)

Set $total\_len_{t^*} = total\_len_{t^*} + \Delta len_{t^*}(r\_real_{t^*}^k)$ and $begin_{t^*}^{k+1} = l_{t^*}^{k(Final)}$.

(* vehicle t' performs the first part of $r\_real_{t'}^k$ *)

Determine the customers served during stage k for vehicle t' by finding the location of t' when its distance traveled reaches the shorter length (the updated $total\_len_{t^*}$).

case 1: t' is located at customer $l_{t'}^k$ or will serve this customer after replenishing.

Set $u_{k+1} = R_{s_{t'}^k}(l_{t'}^k)$, $total\_len_{t'} = total\_len_{t'} + \Delta len_{t'}(subroute(r\_real_{t'}^k, begin_{t'}^k, l_{t'}^k))$, $begin_{t'}^{k+1} = l_{t'}^k$.

case 2: t' is en route from customer $l_{t'}^k$ to customer $j_{t'}^k$.

Set $u_{k+1} = R_{s^k_{t'}}(l^k_{t'}) \setminus \{j^k_{t'}\} = R_{s^k_{t'}-1}(j^k_{t'}; l^k_{t'})$, $total\_len_{t'} = total\_len_{t'} + \Delta len_{t'}(subroute(r\_real^k_{t'}, begin^k_{t'}, j^k_{t'}))$, and $begin^{k+1}_{t'} = j^k_{t'}$.

Go to Step 6.

Step 5: Apply the approximate partial reoptimization to vehicle t^{**}

Find actual route $r\_real^k_{t^{**}}$ for customer set u_k.

Set $u_{k+1} = \phi$, $total\_len_{t''} = total\_len_{t''} + d(l^{k-1\,(Final)}_{t''}, 0)$,

$total\_len_{t^{**}} = total\_len_{t^{**}} + \Delta len_{t^{**}}(r\_real^k_{t^{**}}) + d(l^{k\,(Final)}_{t^{**}}, 0)$.

Go to Step 6.

Step 6: if $u_{k+1} = \phi$, set $k + 1 = K^*$ and return $total\_len_1 + total\_len_2$ as the final travel cost.

else set $k = k + 1$ and return to Step 2.

After we find $r\_real^k_1$ and $r\_real^k_2$ for α_k and $u_k \setminus \alpha_k$ respectively at Step 3.2, the calculation of which vehicle completes its assignment first should be performed immediately. It is based on the cumulative distance, since the distance traveled is proportional to the time. The vehicle that finishes first, denoted t^* (corresponding to $\arg\min_{t\in\{1,2\}}\{total\_len_t + \Delta(r\_real^k_t)\}$), travels the route $r\_real^k_{t^*}$ and serves all its current customers before the next communication. The other vehicle t' travels the first part of its route $r\_real^k_{t'}$. We need to determine which customers it is able to serve. If it is serving customer $l^k_{t'}$, the subroute is $subroute(r\_real^k_{t'}, begin^k_{t'}, l^k_{t'})$; if it is en route to customer $j^k_{t'}$, the subroute is $subroute(r\_real^k_{t'}, begin^k_{t'}, j^k_{t'})$. Its remaining customers, $R_{s^k_{t'}}(l^k_{t'})$ or $R_{s^k_{t'}}(j^k_{t'})$, form the customer set u_{k+1}, which will then be partitioned into α_{k+1} and $u_{k+1} \setminus \alpha_{k+1}$. Steps 2, 3, and 4 repeat until $u_{\hat{k}+1} = \phi$, when the heuristic returns the final cost. We note $\hat{k} = K^* - 1$, the second to last stage, to maintain consistency with previous notation.

The condition $u_{\hat{k}+1} = \phi$ is satisfied in two situations. If $t' = t^*$ (Step 4, is condition is satisfied), i.e., the vehicles complete their assignments simultaneously and there are no remaining customers, the vehicles return to the depot and we add $d(l^{\hat{k}\,(Final)}_1, 0)$ and $d(l^{\hat{k}\,(Final)}_2, 0)$ to the final distances. If the new partition is $(u_{\hat{k}}, \phi)$ or $(\phi, u_{\hat{k}})$ (Steps 2 and 5,else branch), the vehicle with an empty assignment returns to the depot at a cost of $d(l^{\hat{k}-1\,(Final)}_{t''}, 0)$, since it is currently at the final location of stage $\hat{k} - 1$. The other vehicle must first complete its new assignment, and the cost is the sum of $\Delta len_{t^{**}}(r\_real^{\hat{k}}_{t^{**}})$ and $d(l^{\hat{k}\,(Final)}_{t^{**}}, 0)$.

6. Computational study

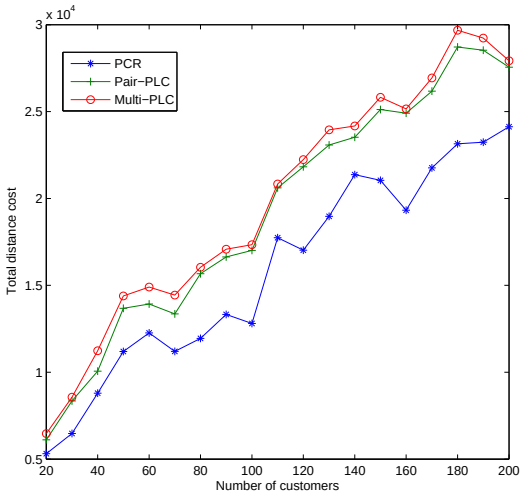
To generate the instances, we use a scheme that is common in VRPSD research, e.g., [8]. The customer locations are randomly generated in a 1,000 by 1,000 square. The depot location is $(0, 0)$ or $(500, 500)$, labeled corner and midpoint respectively. The customer demands are divided into low, medium, and high, corresponding to three discrete uniform random variables with ranges $\{0, \ldots, 4\}$, $\{5, \ldots, 9\}$, and $\{10, \ldots, 14\}$. Each customer $i \in C$ is assigned to one of the three demands with equal probability. Thus, the mean customer demand is $(2 + 7 + 12)/3 = 7$. The value $\bar{f} = \sum_{i=1}^{N} E[\xi]/(2 \times Q)$ [23] measures the expected capacity in use, where $2 \times Q$ indicates that there are two vehicles. Given values for $\bar{f}$ and N, the vehicle capacity Q can be computed by rounding the ratio $7N/(2 \times \bar{f})$ to the nearest

integer. Ten instances are generated for each combination of (i) number of customers: $N = 20, \ldots, 200$, in increments of 10; (ii) depot position: $(0, 0)$ or $(500, 500)$; (iii) $\bar{f}$=1.6 or $\bar{f}$=1.9. There are $10 \times 19 \times 2 \times 2 = 760$ instances.

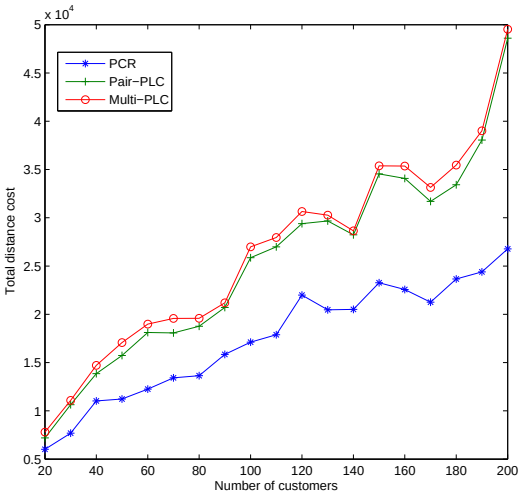
To analyze the PCR, we conduct three experiments. We first compare the PCR with the PLC cooperation strategy. We then make two additional comparisons to evaluate the communication and the partitioning.

6.1. Performance of PCR

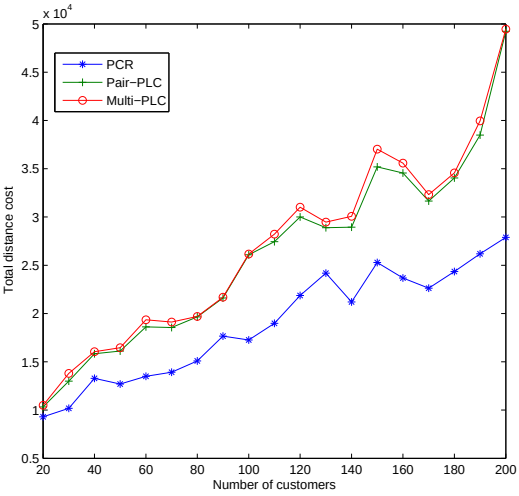
In this section, we compare the PCR solutions to those obtained using the PLC, a recent cooperation technique, in which each pair of vehicles serve customers sequentially in opposite directions, following a fixed route, and if one vehicle fails, the remaining customers are assigned to its partner (see [19]). We set the PLC route to the a priori route generated by RA when $u_0 = C$ (found at Step 1 in Section 5.2). We implement two versions of the PLC. The Pair-PLC has two vehicles, and the Multi-PLC determines the number of vehicles to use to achieve a given service quality (specified in [19] as a probability). The results for PCR, Pair-PLC, and Multi-PLC are given in Figure 3 and Table 1.



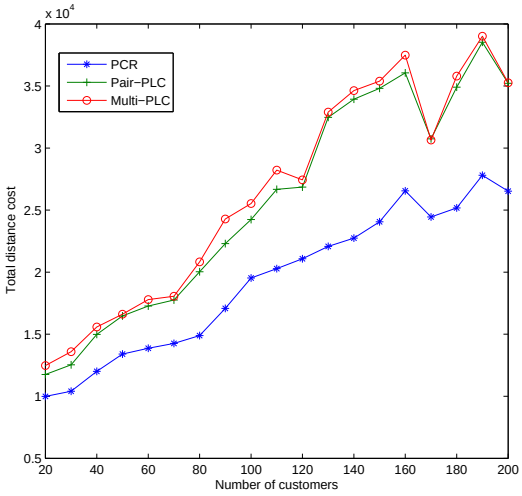
(a) Midpoint instances, $\bar{f} = 1.6$



(b) Midpoint instances, $\bar{f} = 1.9$



(c) Corner instances, $\bar{f} = 1.6$



(d) Corner instances, $\bar{f} = 1.9$

Fig. 3 Performance of PCR versus Pair-PLC and Multi-PLC

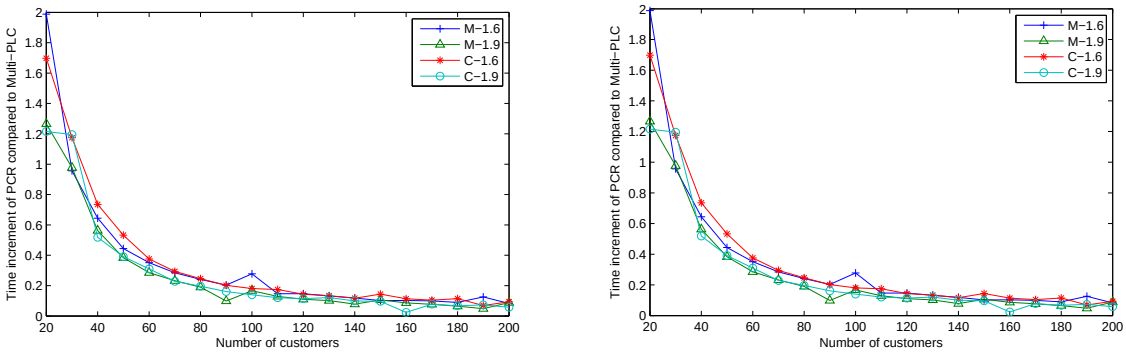
Figure 3 shows the performance of the three approaches for the two depot locations with $\bar{f} = 1.6$ or $\bar{f} = 1.9$. Each data point is an average over 10 instances. The PCR clearly outperforms the other two approaches; Table 1 summarizes the associated cost reductions. It shows that the improvement ranges from 20% to 30%.

Table 1 Percentage improvement in PCR compared to Pair-PLC and Multi-PLC (cost)

# customers	Midpoint depot				Corner depot			
	$\bar{f}$ = 1.6		$\bar{f}$ = 1.9		$\bar{f}$ = 1.6		$\bar{f}$ = 1.9	
	PCR vs. Pair-PLC	PCR vs. Multi-PLC	PCR vs. Pair-PLC	PCR vs. Multi-PLC	PCR vs. Pair-PLC	PCR vs. Multi-PLC	PCR vs. Pair-PLC	PCR vs. Multi-PLC
20	13.04%	17.84%	16.37%	22.75%	9.90%	11.34%	15%	19.89%
30	22.43%	24.39%	27.70%	30.57%	21.66%	26.20%	16.99%	23.45%
40	12.59%	21.73%	20.43%	25.04%	16.12%	17.22%	19.85%	22.94%
50	18.17%	22.22%	28.60%	34.20%	21.17%	22.84%	18.64%	19.32%
60	11.98%	17.74%	32.37%	35.48%	27.52%	30.28%	19.69%	22.03%
70	16.18%	22.45%	25.76%	31.44%	24.96%	27.18%	19.73%	21.06%
80	23.76%	25.53%	27.33%	30.38%	23.18%	23.36%	25.67%	28.51%
90	19.89%	21.97%	23.51%	25.22%	18.29%	18.55%	23.41%	29.67%
100	24.72%	26.13%	33.82%	36.56%	33.85%	34.04%	19.42%	23.48%
110	13.92%	14.84%	33.71%	36.02%	30.83%	32.75%	23.94%	28.13%
120	22.01%	23.45%	25.17%	28.24%	27.11%	29.49%	21.50%	23.16%
130	17.78%	20.78%	31.03%	32.40%	16.28%	17.94%	32%	32.88%
140	9.16%	11.59%	27.38%	28.41%	26.76%	29.50%	32.98%	34.32%
150	16.23%	18.52%	32.65%	34.24%	28.15%	31.71%	30.89%	32.03%
160	22.40%	23.18%	33.77%	36.16%	31.48%	33.45%	26.36%	29.16%
170	16.86%	19.21%	32.94%	35.87%	28.53%	30%	20.46%	20.22%
180	19.39%	22.02%	29.19%	33.27%	28.48%	29.56%	27.88%	29.68%
190	18.56%	20.50%	35.89%	37.46%	31.94%	34.44%	27.83%	28.73%
200	12.43%	13.59%	44.90%	45.93%	43.35%	43.60%	24.68%	24.76%
Ave. Imp.	17.40%	20.40%	30%	33%	26%	28%	23.50%	25.97%

In fact, the PLC is only a cooperation strategy, which relies on a fixed route (the a priori route), that two vehicles sequentially serve customers in the order or the opposite direction of the a priori route, whatever the demands reveal. So, the CPU-time is primarily caused by the computation of the a priori-route. But, our strategy, the PCR, will further improve (re-optimize) the customer visiting sequence. It means that we will use additional time to re-optimize the initial a priori route.

The time spent in our method will cost more time compared to the PLC. But we explore whether the expense of the additional time will bring a reward in saving the total distance cost. Table A.1 (in Appendix A) shows the associated time increments, and Figure 4 indicates the additional time expenditure in performance improvements.



(a) Time increment of PCR vs. Pair-PLC

(b) Time increment of PCR vs. Multi-PLC

Fig. 4 CPU-time increment of PCR vs. Pair-PLC and Multi-PLC

Figure 4(a) and 4(b) show the gaps of CPU-time between the PCR versus the Pair-PLC and Multi-PLC get smaller along with the problem size increase. The gaps decrease from almost 2 times in the smallest size 10 customers to within 6% around 180 customers. It indicates the PCR is more capable in solving larger problem size, with nearly equal time efforts but superior performance in cost savings.

6.2. Characteristics of PCR strategy

The advantages of the PCR are its dynamic visiting sequence and the cooperation via communication and partitioning. In the PLC, task division happens only when the first failure occurs, whereas the PCR has more than one opportunity to update the assignments. We now investigate the performance of the communication and partitioning.

6.2.1. Performance of communication

We first compare a PCR strategy (called Com) with communication to update the initial partition $(\alpha_0, C \setminus \alpha_0)$ and a strategy (called NonCom) without such communication. Table 2 presents the average costs and percentage improvements for Com vs. NonCom, where the average is over 10 instances. The better performance in each case is shown in bold. On average, Com dominates NonCom, e.g., 15845.39 versus 16098.9 for Midpoint with $\bar{f} = 1.6$. Com has a lower cost for 15 of the 19 problems for Midpoint with $\bar{f} = 1.6$. The average improvements are 1.41%, 0.93%, 0.84%, and 1.7% for the four cases, with Com on average about 1.22% better than NonCom.

Table 2 Comparison of cooperation with communication and without communication (cost)

#	Midpoint $\bar{f}$ = 1.6 (cost)			Midpoint $\bar{f}$ = 1.9 (cost)			Corner $\bar{f}$ = 1.6 (cost)			Corner $\bar{f}$ = 1.9 (cost)		
customers	Com	NonCom	Imp.	Com	NonCom	Imp.	Com	NonCom	Imp.	Com	NonCom	Imp.
20	5313.95	**5203.27**	-2.13%	**6023.69**	6408.94	6.01%	**9306.39**	9416.63	1.17%	**9992.34**	10356.33	3.51%
30	**6475.15**	6581.35	1.61%	**7691.39**	7786.35	1.22%	10178.95	**9755.30**	-4.34%	**10405.47**	11113.72	6.37%
40	**8796.19**	9062.48	2.94%	11026.59	**10987.38**	-0.36%	13287.59	**12772.51**	-4.03%	**12009.07**	12422.78	3.33%
50	11190.91	**11118.24**	-0.65%	**11231.17**	11458.98	1.99%	12701.51	**12128.77**	-4.72%	13404.95	**12686.04**	-5.67%
60	12259.69	**11735.63**	-4.47%	**12248.20**	12503.51	2.04%	**13492.57**	13892.67	2.88%	**13869.27**	15134.16	8.36%
70	**11196.91**	11522.48	2.83%	13419.68	**13363.64**	-0.42%	**13918.06**	15168.49	8.24%	**14254.41**	14806.35	3.73%
80	**11945.53**	12362.21	3.37%	**13637.48**	13708.71	0.52%	**15093.68**	15389.79	1.92%	**14890.82**	15295.85	2.65%
90	**13328.09**	13689.56	2.64%	15845.28	**15339.38**	-3.30%	17662.64	**17359.34**	-1.75%	**17082**	17233.32	0.88%
100	**12807.86**	13595.97	5.80%	17118.53	**16788.52**	-1.97%	**17256.84**	18181.44	5.09%	19534.95	**19429.73**	-0.54%
110	**17740.52**	18125.46	2.12%	**17888.56**	18434.66	2.96%	**18982.55**	19458.37	2.45%	**20283.95**	21497.23	5.64%
120	**17021.77**	17122.23	0.59%	21994.80	**21569.19**	-1.97%	**21870.30**	22876.08	4.40%	**21082.75**	21388.18	1.43%
130	**18973.7**	19095.02	0.64%	**20463.13**	21143.62	3.22%	24183.17	**22979.58**	-5.24%	**22080.12**	22256.77	0.79%
140	**21371.28**	21443.07	0.33%	**20506.99**	20597.13	0.44%	**21198.19**	22579.51	6.12%	22745.28	**22657.4**	-0.39%
150	**21041.67**	21422.22	1.78%	**23269.29**	23396.54	0.54%	25287.67	**24484.08**	-3.28%	24055.08	**23994.12**	-0.25%
160	**19325.54**	20270.03	4.66%	**22568.07**	23091.12	2.27%	**23676.75**	24320.66	2.65%	26552	**25797.23**	-2.92%
170	21761.76	**21043.65**	-3.41%	**21251.34**	21909.98	3.01%	**22622.37**	23069.55	1.94%	24449.59	**24350.7**	-0.41%
180	**23147.37**	23602.99	1.93%	23662.68	**23027.62**	-2.76%	**24351.51**	24494.95	0.59%	**25173.61**	26604.55	5.38%
190	**23236.91**	23912.79	2.83%	**24396.97**	25220.36	3.26%	**26190.33**	26351.81	0.61%	**27803.02**	28148.02	1.23%
200	**24127.6**	24970.51	3.38%	**26778.41**	27015.6	0.88%	**27894.02**	28276.91	1.35%	26527.89	**26326.72**	-0.77%
Ave.	**15845.39**	16098.9	1.41%	**17422.22**	17565.85	0.93%	**18902.9**	19102.97	0.84%	**19273.50**	19552.59	1.70%
Superior	15 vs. 4		\	13 vs. 6		\	13 vs. 6		\	12 vs. 7		\

NonCom is the method without following communications to adjust vehicle assignment, the time will be saved for lack of recalculation of the cost-to-go values and adjustment of guidance for vehicle services after each communication interrupts. Table B.1 (in Appendix B) shows the more time spent by Com versus NonCom. On average, the CPU-time increment is $\frac{1}{4} \times (0.2\% + 0.28\% + 0.96\% + 0.66\%) = 0.53\%$, but Com dominates NonCom with average 1.22% in cost saving, so, we can see that the additional time spent in vehicle adjustment by triggering communications will result a valuable improvement in cost reduction.

Another interesting question to consider is whether or not the number of communications influences the quality of the results obtained. Table 4 reports, for each instance size, the average number of times communication occurs between the vehicles. One can see that this number lies between 1 and 4, with larger instances generating more communications. From Tables 2 and 4, it can be observed that a high number of communications does not necessarily entail an improvement in the costs of the solutions obtained. For example, if one considers the Corner instances with $\bar{f} = 1.6$, the average number of communications generated on the instances with 140 and 150 customers is four (results obtained from Table 4). However, an average improvement of 6.12% is observed in the results obtained

for the instances with 140 customers, compared with an average improvement of -3.25% for the instances with 150 customers. Therefore, it is probably the moment at which the vehicles communicate, and not just the number of times the vehicles communicate, that makes the difference in such cases.

6.2.2. Performance of partitioning

To evaluate our partitioning (called App-Par), we compare it with another approach (called Sim-Par) in which we divide each a priori route $r^k = \{l, i_1^k, i_2^k, \ldots, i_{j^*}^k, i_{j^*+1}^k, \ldots, i_n^k, l'\}$ (Step 1 of Section 5.2) at the point

$$j^* = \lfloor \frac{\mid u_k \mid}{2} \rfloor, \quad 0 \leq k \leq K^* - 1. \tag{19}$$

Thus, each vehicle is assigned half of the customers: $r_1^k = \{l, i_1^k, i_2^k, \ldots, i_{\lfloor \frac{|u_k|}{2} \rfloor}^k\}$ and $r_2^k = \{l', i_n^k, i_n^k - 1, \ldots, i_{\lfloor \frac{|u_k|}{2} \rfloor + 1}^k\}$.

Table 3 Comparison of App-Par and Sim-Par (cost)

# customers	Midpoint $\bar{f}$ = 1.6 (cost)			Midpoint $\bar{f}$ = 1.9 (cost)			Corner $\bar{f}$ = 1.6 (cost)			Corner $\bar{f}$ = 1.9 (cost)		
	App-Par	Sim-Par	Imp.	App-Par	Sim-Par	Imp.	App-Par	Sim-Par	Imp.	App-Par	Sim-Par	Imp.
20	**5313.95**	6748.76	21.26%	**6023.69**	6098.52	1.23%	**9306.4**	11971.7	22.26%	9992.34	**9487.95**	-5.32%
30	**6475.15**	6802.22	4.81%	**7691.39**	8337.12	7.75%	**10178.95**	10413.96	2.26%	**10405.47**	12149.79	14.36%
40	**8796.19**	9747.64	9.76%	**11026.6**	11775.44	6.36%	13287.59	**13138.43**	-1.14%	**12009.07**	13430.3	10.58%
50	**11190.91**	11843.38	5.51%	**11231.17**	11817.75	4.96%	**12701.51**	16813.28	24.46%	13404.95	**12775.45**	-4.93%
60	12259.7	**12189.71**	-0.57%	12248.2	**11703.99**	-4.65%	**13492.57**	16553.05	18.49%	**13869.27**	14947.96	7.22%
70	**11196.91**	11385.48	1.66%	13419.68	**13272.79**	-1.10%	**13918.06**	15535.06	10.41%	**14254.41**	14941.35	4.60%
80	**11945.53**	12129.55	1.52%	**13637.48**	14015.17	2.69%	**15093.68**	17364.9	13.08%	**14890.82**	20445.42	27.17%
90	13328.09	**12916.39**	-3.19%	15845.28	**15290.83**	-3.63%	17662.64	**17156.03**	-2.95%	**17082**	17150.64	0.40%
100	**12807.86**	14744.04	13.13%	17118.53	**16768.14**	-2.09%	**17256.84**	19185.6	10.05%	19534.95	**18455.34**	-5.85%
110	**17740.52**	19168.21	7.45%	17888.56	**16503.32**	-8.39%	**18982.55**	19764.67	3.96%	**20283.95**	20665.6	1.85%
120	**17021.77**	18480.11	7.89%	21994.8	**20421.8**	-7.70%	**21870.30**	23852.88	8.31%	21082.75	**20512.68**	-2.78%
130	18973.7	**18225.78**	-4.10%	20463.13	**19712.13**	-3.81%	24183.17	**23470.77**	-3.04%	**22080.12**	24786.06	10.92%
140	**21371.29**	21850.31	2.19%	**20506.99**	21151.39	3.05%	**21198.19**	25166.74	15.77%	22745.28	**22588.61**	-0.69%
150	**21041.67**	21154.30	0.53%	23269.29	**22523.59**	-3.31%	25287.67	**25138.98**	-0.59%	24055.08	**23294.44**	-3.27%
160	**19325.54**	20998.81	7.97%	**22568.07**	24104.95	6.38%	**23676.75**	24872.27	4.81%	**26552**	28230.64	5.95%
170	**21761.76**	22733.84	4.28%	21251.34	**20528.98**	-3.52%	22622.37	**22356.04**	-1.19%	24449.59	**24226.22**	-0.92%
180	**23147.37**	25573.77	9.49%	23662.68	**23580.37**	-0.35%	24351.51	**23454.27**	-3.83%	**25173.61**	27221.38	7.52%
190	**23236.91**	24710.15	5.96%	24396.97	**24177.62**	-0.91%	**26190.33**	29282.02	11%	27803.02	**26933.46**	-3.23%
200	**24127.59**	25268.65	4.52%	**26778.41**	26994.61	0.80%	**27894.02**	29917.73	6.76%	26527.89	**25924.11**	-2.33%
Ave.	**15845.39**	16666.9	4.93%	17422.22	**17304.13**	-0.33%	**18902.9**	20284.65	7.29%	**19273.5**	19903.55	3.17%
Superior	16 vs. 3		\	8 vs. 11		\	13 vs. 6		\	10 vs. 9		\

Table 3 gives the results. App-Par generally performs better, with average improvements of 4.93%, 7.29%, and 3.17% in three cases, but a worse (−0.33%) performance for Midpoint with $\bar{f} = 1.9$. The total average cost reduction is $\frac{1}{4} \times (4.93\% + 7.29\% + 3.17\% - 0.33\%) = 3.96\%$.

Table C.1 (in Appendix C) shows the comparison of CPU-time between App-Par and Sim-Par. The table reveals that the two methods have almost equal performance in time cost. In average 10 of each 19 problems, more than half number of problems, the App-Par saves more time, with average 4.9% ($=\frac{1}{4} \times (9.1\% - 0.01\% + 5.33\% + 5.17\%)$) better than Sim-Par. In addition, App-Par also performs better in cost saving with 3.96% improvement. In general, the App-Par reveals a good performance in both cost saving and time expenditure. This can be explained by the fact that App-Par considers the cost associated to the partitions while Sim-Par follows arbitrary criteria.

7. Conclusions

We have proposed the PCR strategy, an effective cooperation approach for vehicle routing with stochastic demands. It performs better than the PLC cooperation scheme. We have also extended partial reoptimization [8] to the

Table 4. Average number of communications in PCR strategy

# customers	Midpoint		Corner	
	$\bar{f}$ = 1.6	$\bar{f}$ = 1.9	$\bar{f}$ = 1.6	$\bar{f}$ = 1.9
20	2.1	1.9	1.9	1.7
30	1.9	1.7	1.2	2.4
40	1.8	1.9	2.1	2.4
50	2.2	3	2.8	2.1
60	2.8	1	2.8	2
70	2.9	2	2	2.1
80	1.1	1	2.9	3.6
90	1.4	1.4	1.1	1
100	2.7	1	1.9	1
110	3.7	1	1.7	1.9
120	2.9	1	2	2.1
130	2	2	2	3.9
140	3	2.2	4	2.9
150	3	3	4	2.8
160	2.4	2	2	4.6
170	2.8	1	1	3
180	2.7	1.8	1.9	2.4
190	1	2.8	4	1.5
200	2	3.4	3.2	2
Ave.	2.33	1.85	2.34	2.39

paired-vehicle case. In the single-vehicle case, partial reoptimization has been shown to perform better than other reoptimization methods such as the roll-out algorithm and approximate dynamic programming.

Compared with the PLC, the PCR reduces the cost by 20% to 30%. When we compare Com and NonCom we find that the use of communication reduces the cost by an average of 1.22%. When we compare App-Par and Sim-Par we find that our partitioning reduces the cost by an average of 3.96%. We believe that this could increase in contexts where routes length would be highly imbalanced, in such cases the vehicle associated to shorter routes would trigger communication and could support those who have much longer routes.

Future research could explore other partitioning methods. For instance, the cost-to-go values of (13) and (14) could be approximated by $\tilde{v}(l, i_1^k, i_{j^*}^k)$ and $\tilde{v}(l', i_n^k, i_{j^*+1}^k)$ instead of the simplified $dis^0(l, i_1^k, i_{j^*}^k)$ and $dis^1(l', i_n^k, i_{j^*+1}^k)$, but at the expense of computational efficiency. Future work should also focus on clustering the customers, so that the PCR can solve the multivehicle problem one cluster at a time. Partial reoptimization [8] could thus be extended to the multivehicle VRPSD. Finally, investigating other strategies to manage communication between vehicles also appears to be an interesting avenue of research.

Acknowledgements

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under the grants 355401 and ..., by the China Scholarship Council under grant 201206250076 and by the Research Fund for the Doctoral Program of Higher Education of China under grant 20100032110034. This support is gratefully acknowledged.

References

[1] Cordeau J-F, Laporte G, Savelsbergh MW, Vigo D. Vehicle routing. In: Handbooks in Operations Research and Management Science: Transportation. 2006;14:367-428.

[2] Golden BL, Raghavan S, Wasil EA. The Vehicle Routing Problem: Latest Advances and New Challenges. Springer; 2008.

[3] Laporte G. Fifty years of vehicle routing. Transportation Science. 2009;43(4):408-16.

[4] Louveaux F, Labbe M, Laporte G, Tanczos K, Toint P. An introduction to stochastic transportation models. NATO ASI Series F Computer and Systems Sciences. 1998;166:244-63.

[5] Lambert V, Laporte G, Louveaux F. Designing collection routes through bank branches. Computers & Operations Research. 1993;20(7):783-91.

[6] Chepuri K, Homem-de-Mello T. Solving the vehicle routing problem with stochastic demands using the cross-entropy method. Ann. Oper. Res. 2005;134(1):153-81.

[7] Yang W-H, Mathur K, Ballou RH. Stochastic vehicle routing problem with restocking. Transportation Science. 2000;34(1):99-112.

[8] Secomandi N, Margot F. Reoptimization approaches for the vehicle-routing problem with stochastic demands. Operations Research. 2009;57(1):214-30.

[9] Dror M, Laporte G, Trudeau P. Vehicle routing with stochastic demands: Properties and solution frameworks. Transportation Science. 1989;23(3):166-76.

[10] Laporte G, Louveaux FV, Van hamme L. An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. Operations Research. 2002;50(3):415-23.

[11] Psaraftis H. Dynamic vehicle routing: Status and prospects. Ann. Oper. Res. 1995;61(1):143-64.

[12] Bertsimas DJ. A vehicle routing problem with stochastic demand. Operations Research. 1992;40(3):574-85.

[13] Secomandi N. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Computers & Operations Research. 2000;27(11C12):1201-25.

[14] Secomandi N. A rollout policy for the vehicle routing problem with stochastic demands. Operations Research. 2001;49(5):796-802.

[15] Goodson JC, Ohlmann JW, Thomas BW. Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. Operations Research. 2013;61(1):138-54.

[16] Bertsekas DP, Tsitsiklis JN. Neuro-dynamic programming: An overview. Proceedings of the 34th IEEE Conference on Decision and Control 1995. pp. 560-4 vol. 1.

[17] Bertsekas D, Tsitsiklis J, Wu C. Rollout algorithms for combinatorial optimization. Journal of Heuristics. 1997;3(3):245-62.

[18] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. Cambridge University Press; 1998.

[19] Ak A, Erera AL. A paired-vehicle recourse strategy for the vehicle-routing problem with stochastic demands. Transportation Science. 2007;41(2):222-37.

[20] Novoa C, Storer R. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. European Journal of Operational Research. 2009;196(2):509-15.

[21] Rei W, Gendreau M, Soriano P. A hybrid Monte Carlo local branching algorithm for the single vehicle routing problem with stochastic demands. Transportation Science. 2010;44(1):136-46.

[22] Secomandi N. Analysis of a rollout approach to sequencing problems with stochastic routing applications. Journal of Heuristics. 2003;9(4):321-52.

[23] Gendreau M, Laporte G, Séguin R. An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation Science. 1995;29(2):143-55.

Appendix A.

Table A.1 CPU-time increment in PCR Versus Pair-PLC and Multi-PLC

# customers	Midpoint depot				Corner depot			
	$\bar{f}$ = 1.6		$\bar{f}$ = 1.9		$\bar{f}$ = 1.6		$\bar{f}$ = 1.9	
	PCR vs. Pair-PLC	PCR vs. Multi-PLC	PCR vs. Pair-PLC	PCR vs. Multi-PLC	PCR vs. Pair-PLC	PCR vs. Multi-PLC	PCR vs. Pair-PLC	PCR vs. Multi-PLC
20	203%	198.80%	130%	126.60%	173.82%	169.68%	123.56%	121.57%
30	96.39%	95.68%	99.12%	97.68%	119.02%	117.43%	121.58%	119.52%
40	64.84%	64.42%	57.00%	56.28%	74.29%	73.50%	52.53%	51.85%
50	44.68%	44.39%	39.02%	38.38%	54%	53.28%	39.30%	39.14%
60	35.26%	35.11%	28.97%	28.53%	38.05%	37.58%	31.15%	31.03%
70	28.66%	28.50%	23.43%	23.23%	29.64%	29.43%	22.95%	22.85%
80	24.19%	24.11%	19.10%	18.99%	24.71%	24.60%	19.79%	19.66%
90	20.36%	20.31%	9.91%	9.86%	20.28%	20.22%	16.16%	16.12%
100	27.78%	27.76%	16.83%	16.68%	18.22%	18.10%	14.05%	13.95%
110	14.76%	14.68%	13.03%	12.90%	17.59%	17.46%	12.11%	12.06%
120	14.74%	14.70%	11.10%	11.02%	14.41%	14.33%	11.49%	11.40%
130	13.24%	13.21%	10.26%	10.18%	13.45%	13.38%	12.23%	12.10%
140	11.92%	11.89%	7.88%	7.83%	11.74%	11.69%	10.03%	9.96%
150	10.21%	10.17%	10.65%	10.59%	14.47%	14.41%	9.71%	9.63%
160	10.28%	10.24%	8.71%	8.64%	11.43%	11.35%	2.35%	2.34%
170	9.96%	9.92%	7.72%	7.66%	10.48%	10.42%	7.88%	7.83%
180	8.95%	8.92%	6.51%	6.45%	11.50%	11.44%	6.67%	6.66%
190	12.56%	12.55%	4.87%	4.83%	6.96%	6.92%	7.33%	7.30%
200	8.27%	8.23%	8.87%	8.79%	9.42%	9.34%	5.82%	5.78%

Appendix B.

Table B.1 CPU-time increment of cooperation with communication versus without communication

# customers	Midpoint $\bar{f}$ = 1.6 (second)			Midpoint $\bar{f}$ = 1.9 (second)			Corner $\bar{f}$ = 1.6 (second)			Corner $\bar{f}$ = 1.9 (second)		
	Com	NonCom	Time-Incre.	Com	NonCom	Time-Incre.	Com	NonCom	Time-Incre.	Com	NonCom	Time-Incre.
20	1.68	1.65	1.86%	1.35	1.35	0	1.42	1.41	1.09%	1.3	1.29	0.44%
30	4.42	4.42	0.16%	4.52	4.52	0.04%	4.98	4.94	0.67%	5.38	5.15	4.46%
40	19.47	19.47	0	18.29	18.15	0.76%	20.3	19.69	3.09%	19.57	19.55	0.12%
50	41.33	41.33	0.01%	37.97	37.96	0.04%	42.06	40.23	4.54%	40.35	40.21	0.35%
60	76.86	76.86	0	71.01	70.83	0.25%	76	74.72	1.72%	76.12	76.12	0
70	131.27	131.26	0	124.98	124.98	0	131.27	131.27	0	128.07	128.06	0
80	211.27	210.98	0.14%	204.65	204.64	0	214.29	214.28	0	207.34	207.26	0.04%
90	322.41	322.39	0	299.03	298.51	0.18%	327.22	326.55	0.21%	255.81	254.78	0.40%
100	566.65	566.65	0	309.19	309.18	0	312.87	312.6	0.08%	371.27	371.27	0
110	531.53	531.52	0	523.78	520.48	0.63%	544.92	537.39	1.40%	531.79	531.76	0
120	747.16	743.34	0.51%	724.05	724	0	745.64	745.41	0.03%	740.58	736.73	0.52%
130	1010.42	1007.75	0.26%	984.38	980.98	0.35%	1012.92	1007.89	0.50%	1020.64	998.92	2.17%
140	1333.61	1333.42	0.01%	1286.06	1285.17	0.07%	1332.09	1332.04	0	1334.38	1324.14	0.77%
150	1750.69	1750.5	0.01%	1671.49	1664.35	0.43%	1729.23	1714.48	0.86%	2023.85	2002.03	1.09%
160	2548.04	2540.91	0.28%	2743.7	2723.77	0.73%	2812.15	2785.81	0.95%	6896.25	6893.3	0.04%
170	3463.02	3453.28	0.28%	3383.07	3364.22	0.56%	3469.79	3438.36	0.91%	3438.4	3413.99	0.72%
180	4268.89	4267.5	0.03%	4239.83	4208.07	0.75%	4438.82	4392.97	1.04%	4012.5	4012.26	0
190	5542.74	5533.48	0.17%	7302.9	7273.9	0.40%	7448.65	7404.71	0.59%	3633.48	3605.53	0.78%
200	4319.16	4318.85	0	5307.41	5305.33	0.04%	5334.41	5306.76	0.52%	7689.34	7645.02	0.58%
Ave.	\	\	0.20%	\	\	0.28%	\	\	0.96%	\	\	0.66%

Appendix C.

Table C.1 CPU-time increment of App-Par versus Sim-Par

# customers	Midpoint $\bar{f}$ = 1.6 (second)			Midpoint $\bar{f}$ = 1.9 (second)			Corner $\bar{f}$ = 1.6 (second)			Corner $\bar{f}$ = 1.9 (second)		
	App-Par	Sim-Par	Time-Incre.	App-Par	Sim-Par	Time-Incre.	App-Par	Sim-Par	Time-Incre.	App-Par	Sim-Par	Time-Incre.
20	1.68	**1.31**	-28.24%	1.35	**1.2**	13.04%	1.42	**1.24**	14.35%	1.3	**1.2**	8.13%
30	4.42	**4.14**	-6.78%	4.52	**4.05**	11.84%	4.98	**4.16**	19.55%	5.38	**4.48**	20.20%
40	19.47	**12.65**	-53.94%	18.29	**15.96**	14.57%	20.3	**12.64**	60.57%	19.57	**14.94**	31%
50	41.33	**32.22**	-28.31%	37.97	**36.56**	3.86%	42.06	**34.48**	21.97%	40.35	**31.66**	27.43%
60	76.86	**66.9**	-14.89%	71	**70.64**	0.52%	76	**67.72**	12.23%	76.12	**66.77**	14%
70	131.27	**128.67**	-2.02%	**124.98**	125.3	-0.25%	**131.27**	132.06	-0.60%	**128.07**	132.12	-3.06%
80	**211.27**	242.17	12.76%	204.65	**203.97**	0.34%	214.3	**205.53**	4.26%	207.34	**205.71**	0.80%
90	322.41	**316.54**	-1.86%	**299.03**	314.97	-5.06%	327.22	**318.66**	2.68%	**255.81**	310.95	-17.73%
100	566.65	**331.99**	-70.69%	**309.19**	321.4	-3.80%	**312.87**	461.55	-32.20%	**371.27**	453.74	-18.17%
110	**531.53**	665.34	20.11%	**523.78**	604.59	-13.37%	**544.92**	663.68	-17.89%	**531.79**	650.5	-18.25%
120	**747.16**	911.85	18.06%	**724.05**	749.72	-3.42%	**745.64**	913.49	-18.37%	**740.58**	894.6	-17.21%
130	**1010.42**	1337.29	24.44%	**984.38**	1044.34	-5.74%	**1012.92**	1372.43	-26.20%	**1020.64**	2010.8	-49.24%
140	**1333.61**	1960.84	31.99%	**1286.06**	1455.86	-11.66%	**1332.09**	2515.86	-47.05%	**1334.38**	3194.54	-58.23%
150	**1750.69**	4204.13	58.36%	**1671.49**	1689.8	-1.08%	**1729.23**	4199.3	-58.82%	2023.85	**1801.12**	12.37%
160	**2548.04**	2743.16	7.11%	2743.7	**2722.03**	0.80%	**2812.15**	3222.36	-12.73%	3121.25	**2732.66**	14.22%
170	**3463.02**	3651.73	5.17%	3383.07	**3370.42**	0.38%	**3469.79**	4522.42	-23.28%	**3438.4**	3813.89	-9.85%
180	4268.89	**3775.11**	-13.08%	**4239.83**	4250.87	-0.26%	4438.82	**4227.52**	5%	**4012.5**	4212.92	-4.76%
190	5542.74	**5203.03**	-6.5%	**7302.9**	7313.06	-0.14%	7448.65	**5971.73**	24.73%	**3633.48**	5728.79	-36.58%
200	**4319.16**	6266.84	31.08%	**5307.41**	5322.26	-0.28%	**5334.41**	7553.98	-29.38%	7689.34	**7200.24**	6.79%
Ave.	\	\	-9.10%	\	\	0.01%	\	\	-5.33%	\	\	-5.17%
Superior	9 vs. 10		\	11 vs. 8		\	10 vs. 9		\	10 vs. 9		\