A robust optimization approach for the vehicle routing problem with cross-docking under demand uncertainty

Highlights

A Robust Optimization Approach for the Vehicle Routing Problem with Cross-Docking Under Demand Uncertainty

- Vehicle routing problem with cross-docking under demand uncertainty.
- Nominal and robust mathematical formulations.
- ALNS algorithm embedding efficient feasibility checks for the robust constraints.
- Extensive computational study proving the effectiveness of the proposed algorithm.
- Price of robustness and managerial insights.

A Robust Optimization Approach for the Vehicle Routing Problem with Cross-Docking Under Demand Uncertainty

Vincent F. Yu[a,b], Pham Tuan Anh[a,*] and Roberto Baldacci[c]

[a]*Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan*
[b]*Center for Cyber-Physical System Innovation, National Taiwan University of Science and Technology, Taipei, Taiwan*
[c]*Engineering Management and Decision Sciences, College of Science and Engineering, Hamad Bin Khalifa University, Doha P.O. Box 34110, Qatar*

ARTICLE INFO

Keywords:
Uncertainty modelling
Robust optimization
Uncertain demand
Vehicle routing problem with cross-docking
Adaptive large neighborhood search.

ABSTRACT

This research addresses the Vehicle Routing Problem with Cross-docking under Demand Uncertainty (VRPCD-DU) where a set of homogeneous vehicles is used to transport orders from the suppliers to the corresponding customers via a cross-dock. VRPCD-DU considers customer demand volumes as random variables and determines a minimum cost delivery plan that is feasible for all anticipated demand realizations. A robust optimization counterpart of a deterministic VRPCD formulation is derived where support for the demand is a polyhedral set. The robust formulation can only be used to solve small VRPCD-DU instances, and an effective adaptive large neighborhood search (ALNS) algorithm is proposed for solving large instances. Extensive numerical experiments are conducted on benchmark sets for both VRPCD and VRPCD-DU. The results show that the ALNS algorithm computes new best-known solutions of the VRPCD benchmark instances. Moreover, demand uncertainty is extensively analyzed by investigating managerial insights. The price of robustness (PoR) is compared by considering various budget sets and alternative partitioning methods for the customers (i.e., “random" and “clustered"). The findings imply that a higher probability of capacity constraints’ violation is generally observed for the suppliers and that customer partitioning methods share similar PoR.

*Corresponding author
vincent@mail.ntust.edu.tw (V.F. Yu); phamtuananhise@gmail.com (P.T. Anh); rbaldacci@hbku.edu.qa (R. Baldacci)
ORCID(s):

A Robust Optimization Approach for the Vehicle Routing Problem with Cross-Docking Under Demand Uncertainty

ARTICLE INFO

Keywords:
Uncertainty modelling
Robust optimization
Uncertain demand
Vehicle routing problem with cross-docking
Adaptive large neighborhood search.

ABSTRACT

This research addresses the Vehicle Routing Problem with Cross-docking under Demand Uncertainty (VRPCD-DU) where a set of homogeneous vehicles is used to transport orders from the suppliers to the corresponding customers via a cross-dock. VRPCD-DU considers customer demand volumes as random variables and determines a minimum cost delivery plan that is feasible for all anticipated demand realizations. A robust optimization counterpart of a deterministic VRPCD formulation is derived where support for the demand is a polyhedral set. The robust formulation can only be used to solve small VRPCD-DU instances, and an effective adaptive large neighborhood search (ALNS) algorithm is proposed for solving large instances. Extensive numerical experiments are conducted on benchmark sets for both VRPCD and VRPCD-DU. The results show that the ALNS algorithm computes new best-known solutions of the VRPCD benchmark instances. Moreover, demand uncertainty is extensively analyzed by investigating managerial insights. The price of robustness (PoR) is compared by considering various budget sets and alternative partitioning methods for the customers (i.e., “random" and “clustered"). The findings imply that a higher probability of capacity constraints’ violation is generally observed for the suppliers and that customer partitioning methods share similar PoR.

1. Introduction

The Vehicle Routing Problem (VRP), first introduced by Dantzig and Ramser (1959) for truck dispatching, is a widely known combinatorial optimization problem that has attracted the interest of many researchers (Braekers et al., 2016; Toth and Vigo, 2014). Among the variants of VRP is the Vehicle Routing Problem with Cross-Docking (VRPCD), where products are transported from suppliers to customers via an intermediate facility called a cross-dock (Wen et al., 2009). VRPCD was first addressed by Lee et al. (2006) to design vehicle routes for inbound and outbound trucks at minimum total operational cost. An important assumption is that all input parameters (customer demands and travel times) are deterministic or known *a priori*. In practice, input data are affected by uncertainty, and solutions of deterministic models can be highly impracticable in many situations (Luce and Raiffa, 1954; Taş et al., 2013; Kiani Mavi et al., 2020). Moreover, uncertain factors reduce the efficiency of optimized operations in cross-docking. Thus, stakeholders in cross-docking networks need to adapt to those factors (Kiani Mavi et al., 2020).

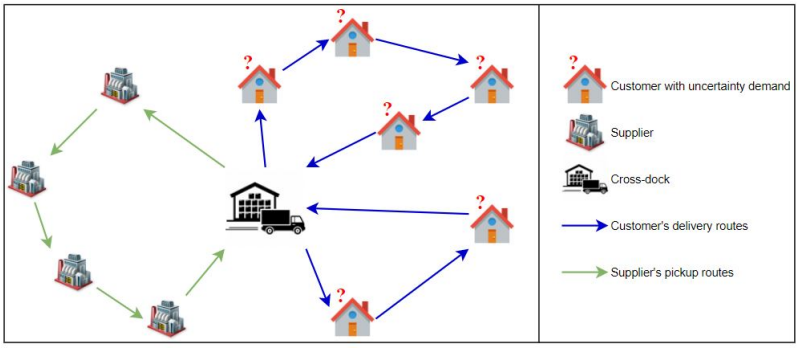


Figure 1: Schematic representation of VRPCD under demand uncertainty

To deal with uncertainties in optimization problems, the literature presents two of the main solution approaches used: stochastic programming (SP) and robust optimization (RO). Stochastic programming is applied when historical

ORCID(s):

data are available to define the probability distributions of uncertain factors. It aims to optimize the expected outcomes for all scenarios (Birge and Louveaux, 2011). Two main approaches of SP are chance-constrained programming (CCP) and stochastic programming with recourse (SPR) (Ordóñez, 2010).

Even though SP has been used with success for many applications, it has two main drawbacks. First, its key assumption is that the probability distributions of uncertain data must be known precisely. Second, computational tractability is reduced due to the curse of possible outcomes (Gounaris et al., 2013; Solano-Charris et al., 2015). RO is an alternative approach used to avoid SPs drawbacks to overcome the lack of historical data on real problems.

In RO the more additional information over the uncertainty that is available, the less conservative the solution that is produced (Bertsimas et al., 2011; Munari et al., 2019). Decision-makers in RO can define the right strategy toward uncertainty to balance the robustness of solutions and the performance of solutions in terms of benefits or costs. Additional information is addressed by predefined *uncertainty sets* such as ellipsoids, polyhedrons, and boxes (Ben-Tal and Nemirovski, 2002; Bertsimas and Sim, 2003). Despite the complexity of defining the shape of an uncertainty set, RO is computationally tractable in that the resulting optimization problems can be solved in reasonable computational times (Jaillet et al., 2016).

In this paper we address VRPCD under demand volume uncertainty (VRPCD-DU) (see Figure 1 for a schematic representation of the problem). The contributions of this paper are summarized as follows.

- To the best of our knowledge, this paper is the first to investigate demand uncertainty in the context of VRPCD.
- We derive the robust counterpart of a deterministic model for VRPCD based on Miller-Tucker-Zemlin (MTZ) constraints. To model uncertainty, we adopt polyhedral sets such as the budget set.
- To solve large-size VRPCD-DU instances, we design an effective adaptive large neighborhood search (ALNS) metaheuristic based on infeasible solutions and on new destroy and repair operators. In particular, ALNS relies on an efficient procedure used to check the feasibility of the robust constraints.
- We consider both VRPCD benchmark instances taken from the literature and newly generated VRPCD-DU instances and report extensive computational experiments about the robust model and the ALNS algorithm. In particular, ALNS is used to solve both VRPCD and VRPCD-DU instances. The results show that the ALNS algorithm outperforms the state-of-the-art algorithms for VRPCD and that high quality solutions are computed for VRPCD-DU.
- We report sensitive analyses on different VRPCD parameters and offer some managerial insights. In particular, we investigate the *price of robustness* of the computed solutions.

The remainder of this paper is structured as follows. Section 2 gives a literature review on related works. Section 3 presents the deterministic VRPCD model, while Section 4 formulates its robust counterparts. Section 5 describes the ALNS algorithm including details of the relevant operators. Section 6 presents extensive computational experiments. Section 7 concludes the paper and points out future research directions.

2. Related work

In this section we review the literature related to VRPCD and VRPCD-DU. Below, Section 2.1 states the literature on VRPs with cross-docking, whereas Section 2.2 gives an overview on RO approaches for VRPs.

2.1. VRPs with cross-docking

For a thorough review of the cross-docking concept and an extensive review of the literature up to 2012, the reader may refer to the review of Van Belle et al. (2012).

VRPCD was introduced by Lee et al. (2006) as a VRP variant with a cross-docking strategy in a supply chain network consisting of suppliers, customers, and a cross-dock. The model assumes that all vehicles in the delivery process arrive at the cross-dock simultaneously. A mixed integer linear programming (MILP) model and a tabu-search (TS) algorithm were developed to solve self-generated benchmark instances. Several researchers have proposed effective solution methods such as modified-TS (Liao et al., 2010), Simulated Annealing (SA) (Yu et al., 2016), Adaptive Neighborhood Simulated Annealing (ANSA) (Yu et al., 2021), and two-phase matheuristic based on ALNS (Gunawan et al., 2021).

Table 1
Overview of the related literature on VRPCD

Reference	Features of the cross-docking problems	Objective function	Uncertain factor	Approach
Lee et al. (2006)	Simultaneously arrival and departure	Travel + Operational costs	None	TS algorithm
Wen et al. (2009)	Relaxing simultaneous arrival and departure + Delivery time window	Travel cost	None	TS algorithm
Yu et al. (2016)	Open routes	Travel + Operational costs	None	SA algorithm
Mousavi and Vahdani (2017)	Multiple cross-docks + Facility location decisions	Holding inventories + travel + Fixed opening cost	Travel cost + Vehicle capacity	SAICA
Rahbari et al. (2019)	Heterogeneous fleet + Delivery time window	Travel + penalty + holding costs and Freshness of products	Travel time + Freshness time	RO model
Yu et al. (2021)	Heterogeneous fleet + Multiple cross-docks	Travel + Operational costs	None	ANSA
This paper	Simultaneous arrival and departure	Travel + Operational costs	Demand	RO model modified ALNS

Wen et al. (2009) proposed VRPCD with time windows (VRPCDTW), which is an extension of VRPCD by considering time windows and relaxing the simultaneous arrival constraints. A MILP model and a TS algorithm were set up to solve self-generated benchmark instances. Solutions were then improved by later algorithms such as TS algorithm (Tarantilis, 2013), Genetic Algorithm (GA) (Touihri et al., 2016), Iterated Local Search (ILS) (Morais et al., 2014), and matheuristic (Grangier et al., 2016).

Open VRPCD (OVRPCD) was introduced by Yu et al. (2016) in which the pickup routes start from suppliers and end at the cross-dock. The delivery routes start at the cross-dock and end at the customers. The routes do not form a closed-loop.

The literature has seldom addressed VRPs with cross-docking under uncertainty. Mousavi et al. (2013) targeted VRPCD under an uncertain environment including fixed costs, travel costs, and the cross-dock's capacity. A fuzzy possibilistic two-phase approach was proposed to address the problem. A numerical experiment was conducted to show the applicability and suitability of the approach, but the model can only treat a single uncertain factor at a time. To tackle this drawback, Mousavi et al. (2014) introduced a hybrid fuzzy possibilisticstochastic programming solution approach to simultaneously consider two uncertain factors in the model. Mousavi and Vahdani (2017) employed an RO approach to address the same problem. A self-adaptive imperialist competitive algorithm (SAICA) was developed to solve the problem. Results indicated the applicability and capability of RO in solving real-world problems under uncertainty.

Rahbari and Nasiri (2017) designed RO for VRPCD such that the routing plans are immune to loading and unloading time uncertainties. The interval-polyhedral (Bertsimas and Sim, 2004) was adopted to reformulate a robust version that minimizes travel, cross-docking, inventory holding, and penalty costs. Results exposed the effects of the conservatism and variability levels on the robustness of solutions. Rahbari et al. (2019) mentioned two uncertainties including travel time and freshness-life, which are typical factors of perishable products. They formulated two robust models looking at the uncertain environment. The L1 metric method was proposed to solve bi-objective functions in terms of minimizing the total cost (including earliness and tardiness penalties, inventory holding, and travel costs) and maximizing the total weighted freshness of perishable products. Seyedhoseini et al. (2015) offered a queuing theory to deal with demand and service time following Poisson and Exponential distributions, respectively.

Table 1 summarizes the features of the problems studied in the literature and closely related to VRPCD. In particular, the table shows their characteristics regarding cross-docking, objective functions, uncertainty factors, and solution approaches. Our paper is the first to address vehicle routing with cross-docking under demand uncertainty.

In this paper we propose an ALNS algorithm to solve VRPCD and its uncertainty variant. ALNS has been successfully implemented to handle many VRP extensions such as pickup and delivery problems (Ropke and Pisinger, 2006; Masson et al., 2013), routing with synchromodal constraint problems (Grangier et al., 2016; Zhang et al., 2022), and two echelon vehicle routing problems (2E-VRP) (Hemmelmayr et al., 2012). A thorough review of the literature about ALNS algorithms can be found in Pisinger and Ropke (2019).

Table 2
Overview of the related literature on the robust optimization of VRP variants under uncertain factor

Research	VRP variant	Objective	Uncertain factor				Support uncertain set				Approach
			(1)	(2)	(3)	Other	(4)	(5)	(6)	Other	
Sungur et al. (2008)	CVRP	Travel costs	✓	-	-		✓				RO model
Ordóñez (2010)	VRP	Travel costs	✓	✓	✓		✓	✓	✓		Outline of RO
Gounaris et al. (2013)	CVRP	Travel costs	✓	-	-		✓	✓	-		AMP
Gounaris et al. (2016)	CVRP	Travel costs	✓	-	-		✓		-		RO model
Lee et al. (2012)	VRP-D	Travel costs	✓		-		✓	-	-		DW decomposition
Solano-Charris et al. (2015)	CVRP	Travel costs	-	-	✓		-	-	-	Discrete	Local search-based
Eufinger et al. (2020)	CVRP	Travel costs	-	-	✓		✓	✓	-		K-adaptability concept
Agra et al. (2012)	VRPTW	Travel costs	-	✓	-		✓	-	-		RO model
Agra et al. (2013)	VRPTW	Travel costs	-	✓	-		✓	-	-		$\mathcal{T}$-RI and $\mathcal{T}$-PI formulations
De La Vega et al. (2019)	VRPTW	Travel costs + Number of routes and deliverymen	✓	-	-		✓	-	-		Solomons heuristics I1
Hu et al. (2018)	VRPTW	Travel costs + Number of vehicles (routes)	✓	✓	-		✓	✓	-		Two-phase algorithm based on AVNS
Lu and Gzara (2019)	VRPTW	Travel costs	✓	-	-		✓	✓	-		BPC method
Munari et al. (2019)	VRPTW	Travel costs	✓	✓	-		✓	✓	-		BPC method
Santos et al. (2020)	VRPSB	Travel costs minus revenues from backhauls	-	-	-	Revenues	✓	✓	-		ALNS
Mousavi and Vahdani (2017)	VRPCD	Holding inventories + travel + Fixed opening cost	-	-	✓	Capacity	-	-	✓		SAICA
Rahbari et al. (2019)	VRPCD	Travel + penalty + holding costs and Freshness of products	-	✓	-	Freshness	✓	✓	-		RO model
Bartolini et al. (2021)	TSPTW	Travel costs	-	✓	-		-	-	-	Knapsack-constrained	An exact solution based on CG and DP
This paper	VRPCD	Travel + Operational costs	✓	-	-		✓	✓	-		RO model, modified ALNS

Note: (1) Demand uncertainty, (2) Time uncertainty, (3) Cost uncertainty, (4) Polyhedral, (5) Box, (6) Ellipsoidal.

2.2. VRPs under uncertainty

Uncertainty in VRPs has been addressed by several works in the literature, and RO has been adopted as one of the main optimization tools to tackle this class of very challenging problems. Data uncertainty relates to different practical aspects, such as customer demands, travel times, and costs. Table 2 gives an overview of the related works. Below we focus on the literature dealing with demand uncertainty, being closely related to VRPCD considered herein.

Sungur et al. (2008) proposed the first version of RO, implemented to CVRP under demand uncertainty and by minimizing travel costs. The MTZ constraints in the two-index deterministic CVRP model were modified into three robust versions to illustrate demand uncertainty based on the convex hull, box, and ellipsoidal sets, in which the nominal demand in deterministic CVRP exhibits variations with the predefined uncertainty sets. To avoid unmet demand, the worst-case realization is determined to ensure the feasibility of the vehicles loading. The robust CVRP reveals overly conservative solutions in comparison with the stochastic approaches, however, the robust version is more practical and traceable than the stochastic models. Motivated by the above study, Ordóñez (2010) took more uncertain factors (i.e., uncertain demand, time, costs, customers) into VRP. The author constructed a clear outline of RO methodology for VRPs applications under a specific uncertainty (for example, routing in large-scale emergencies and courier delivery problems).

RO more interestingly has been applied to treat CVRP under demand uncertainty (RCVRP) throughout a series of two papers. Derived from the deterministic CVRP formulation, Gounaris et al. (2013) formed robust counterparts based on well-known typical formulations of VRP including two-index vehicle flow, MTZ formulation, one-commodity flow, and two-commodity flow. The RO approach guarantees feasibility for solutions, and even the worst-case of customer demand is realized. Robustness rounded capacity inequalities (RCI) were proposed as cuts in a branch and cut to solve the problem. Due to frequently checking for violations of constraints, budget and factor supports were introduced to alleviate the computational time in a linear form. Gounaris et al. (2016) first introduced an adaptive memory programming (AMP) approach to solve RCVRP. A well-performing algorithm has been proven, whereby new best solutions of 123 out of 180 instances were found from small-, medium-, and large-sized instances in that previous study.

Other extensions of CVRP under uncertainty have been found. First, Lee et al. (2012) studied the RO approach to investigate CVRP with deadlines (VRP-D) with uncertain travel time/demand. The Dantzig-Wolfe (DW) decomposition encapsulates the problem into sub-problems, where each sub-problem is solved by dynamic programming (DP),

but the method cannot directly apply to VRPTW applications. Adulyasak and Jaillet (2016) considered travel time uncertainty in a VRP-D model and developed a branch-and-cut framework for stochastic and robust approaches. While the SP model minimizes the expected number of deadline violations based on the predefined probability, RO ensures feasibility under a certain family of distributions when information about SP is unknown. Solano-Charris et al. (2015) considered each arc cost by a set of discrete scenarios. MILP and local search-based metaheuristics were proposed to handle this problem. Eufinger et al. (2020) set up a min-max-min-robust approach to deal with travel cost uncertainty. The concept of K-adaptability was proposed to obtain K feasible solutions as the robust solution candidates.

3. The deterministic VRPCD: problem definition and mathematical formulation

We consider a cross-docking network including a set of suppliers S, a set of customers C, a single cross-dock represented by node 0, and a homogeneous fleet of vehicles V with a capacity equal to Q. We extend VRPCD, investigated by Lee et al. (2006), to consider multiple products for customer demand and assume that each supplier provides one kind of products. We denote $P = S$ as the set of products.

The problem is defined on an undirected graph $G = G' \cup G''$ consisting of two subgraphs associated with the *supplier* network $G' = (S^+, A')$ and the *customer* network $G'' = (C^+, A'')$. Let two additional sets be $S^+ = S \cup \{0\}$ and $C^+ = C \cup \{0\}$. For arcs connecting two nodes in the network, set $A' = \{(i,j)|i,j \in S^+, i \neq j\}$ denotes the set of arcs between suppliers/cross-dock, and set $A'' = \{(i,j)|i,j \in C^+, i \neq j\}$ denotes the set of arcs between customers/cross-dock. Each customer requests the demand quantities of multiple products, $\hat{d}_{ip} \geq 0$ for all $i \in C$ and $p \in P$. We assume that each supplier $j \in S$ provides a non-negative supply quantity s_j for only one type of product $j = p$, and that it has enough capacity to cover all customer demand; i.e., $s_j = \sum_{i \in C} \hat{d}_{ij}$ for all $j \in S$.

The arcs of the network G are associated with both travel times and travel costs. The travel time associated with the sets of arcs is defined by t'_{ij} for all $(i,j) \in A'$ and t''_{ij} for all $(i,j) \in A''$. Similarly, travel cost associated with the sets of arcs is also defined by c'_{ij} for all $(i,j) \in A'$ and c''_{ij} for all $(i,j) \in A''$. A fixed operating cost F is also associated with each vehicle. We also assume that the total duration of the pickup and delivery routes must be limited by a maximum duration limit T_{max}.

We define a supplier (customer) route to be a simple cycle in G' (G") starting at depot 0, visiting a subset of the set of suppliers S (customers C), and ending at the depot 0. The total demand collected (delivered) by a supplier (customer) route must be less than or equal to the vehicle capacity Q. As previously stated, we assume that the total demand required by the customers is equal to the total demand available at the suppliers.

We assume that the supplier routes are first consolidated at the cross-dock, and that the customer routes are executed once the consolidation process is completed. As also assumed by Lee et al. (2006), the time of the operations at the cross-dock is not considered in VRPCD.

VRPCD consists of designing a set of supplier routes and a set of customer routes of minimum total fixed and travel cost such that:

- Each supplier (customer) is visited by exactly one supplier (customer) route.
- Each vehicle performs at most one supplier (customer) route.
- The total duration of the supplier and customer routes is less than or equal to the maximum duration T_{max}.

The mathematical formulation uses two sets of decision variables: x'_{ijv} for all $(i,j) \in A'$ and $v \in V$ and x''_{ijv} for all $(i,j) \in A''$ and $v \in V$. Let x'_{ijv} be equal to 1 if arc $(i,j) \in A'$ is traversed by vehicle v and 0 otherwise. Similarly, let x''_{ijv} be equal to 1 if arc $(i,j) \in A''$ is traversed by vehicle v and 0 otherwise. Variables x' and x'' describe suppliers' and customers' routes, respectively. In addition, non-negative decision variables T_{SP} and T_{CD} represent the total duration of the suppliers' and customers' routes, respectively. Finally, loading variables define the amount of cumulative load quantity of the vehicle upon suppliers u'_i for all $i \in S$ and the amount of cumulative unload quantity of the vehicle upon customers u''_i for all $i \in C$. The mathematical formulation is as follows.

$$\text{Minimize} \sum_{(i,j)\in A'} \sum_{v\in V} c'_{ij} x'_{ijv} + \sum_{(i,j)\in A''} \sum_{v\in V} c''_{ij} x''_{ijv} + \sum_{i\in S} \sum_{v\in V} F x'_{0iv} + \sum_{i\in C} \sum_{v\in V} F x''_{0iv} \tag{1}$$

Subject to:

$$\sum_{i \in S^+ \setminus \{j\}} \sum_{v \in V} x'_{ijv} = 1 \qquad \forall\, j \in S \tag{2}$$

$$\sum_{i \in C^+ \setminus \{j\}} \sum_{v \in V} x''_{ijv} = 1 \qquad \forall\, j \in C \tag{3}$$

$$\sum_{i \in S^+ \setminus \{k\}} x'_{ikv} = \sum_{j \in S^+ \setminus \{k\}} x'_{kjv} \qquad \forall\, v \in V, k \in S^+ \tag{4}$$

$$\sum_{i \in C^+ \setminus \{k\}} x''_{ikv} = \sum_{j \in C^+ \setminus \{k\}} x''_{kjv} \qquad \forall\, v \in V, k \in C^+ \tag{5}$$

$$\sum_{j \in S} x'_{0jv} \leq 1 \qquad \forall\, v \in V \tag{6}$$

$$\sum_{j \in C} x''_{0jv} \leq 1 \qquad \forall\, v \in V \tag{7}$$

$$\sum_{i \in S} x'_{0iv} = \sum_{j \in S} x'_{j0v} \qquad \forall\, v \in V \tag{8}$$

$$\sum_{i \in C} x''_{0iv} = \sum_{j \in C} x''_{j0v} \qquad \forall\, v \in V \tag{9}$$

$$\sum_{(i,j) \in A'} t'_{ij} x'_{ijv} \leq T_{SP} \qquad \forall\, v \in V \tag{10}$$

$$\sum_{(i,j) \in A''} t''_{ij} x''_{ijv} \leq T_{CD} \qquad \forall\, v \in V \tag{11}$$

$$u'_j - u'_i + Q(1 - \sum_{v \in V} x'_{ijv}) \geq s_j \qquad \forall\, (i,j) \in A', j \neq 0 \tag{12}$$

$$u'_i \leq Q \qquad \forall\, i \in S^+ \tag{13}$$

$$u''_j - u''_i + Q(1 - \sum_{v \in V} x''_{ijv}) \geq \sum_{p \in P} \hat{d}_{jp} \qquad \forall\, (i,j) \in A'', j \neq 0 \tag{14}$$

$$u''_i \leq Q \qquad \forall\, i \in C^+ \tag{15}$$

$$T_{SP} + T_{CD} \leq T_{max} \tag{16}$$

The objective function (1) states to minimize the total cost comprising four terms: the first two terms determine the travel costs of the supplier pickup process and the customer delivery process, and the last two terms account for the operating costs associated with the vehicles used in the solution. Constraint (2) (resp. (3)) ensures that each supplier (resp. customer) is served by exactly one route. Constraints (4) and (5) are flow conservation constraints imposing the structure of suppliers' and customers' routes, respectively. Constraint (6) (resp. (7)) imposes that a vehicle is used at most once for servicing the suppliers (resp. customers). Each vehicle used must start or end at the cross-dock as stated by Constraints (8) and (9). Constraints (10) and (11) define the maximum travel time of all vehicles in the supplier process and the customer process routes, respectively. Constraints (12) to (15) are MTZ constraints and avoid subtours in solution. For the supplier routes, Constraint (12) tracks the cumulative load quantity of each vehicle in the supplier process. Constraint (13) prevents the loading from exceeding the vehicle capacity. Similarly, Constraints (14) and (15) impose the vehicle capacity requirements of the customer routes. Finally, Constraint (16) imposes that the total duration of the supplier and customer routes is less than or equal to the maximum duration T_{max}.

4. Modeling VRPCD-DU: robust counterpart of the deterministic VRPCD formulation

In this section we derive a robust counterpart of the deterministic VRPCD formulation described in the previous section. Contrary to the deterministic VRPCD model, in VRPCD-DU the customer demands are no longer determined as certain values. Instead, the demand volume is the uncertainty parameter following a budget uncertainty set $\mathcal{U}(\mathcal{D})$.

The section is organized as follows. Section 4.1 defines the budget uncertainty set $\mathcal{U}(\mathcal{D})$. Section 4.2 describes the robust formulation based on the support $\mathcal{U}(\mathcal{D})$. Section 4.3 illustrates how the budget uncertainty set can be efficiently handled in the context of the ALNS algorithm proposed in this work.

4.1. Budget uncertainty set

In this work the demand uncertainty set is represented by a *budget uncertainty set* that has been proposed by Gounaris et al. (2013) for CVRP. The budget uncertainty set is a polyhedral set defined as follows:

$$\mathcal{U}(\mathcal{D}) = \{\boldsymbol{d} \in \mathbb{R}_{+}^{|C|\times|P|} : \boldsymbol{d} \in \left[\underline{\boldsymbol{d}}, \bar{\boldsymbol{d}}\right]^{|C|\times|P|} \text{ and } \sum_{i\in\mathcal{B}_l} \boldsymbol{d}_i \leq \boldsymbol{B_l} \; \forall \, l \in \mathcal{L}\}. \tag{17}$$

Let $\mathcal{L} = \{1, \ldots, L\}$ be the set of budget uncertainty sets. We distribute the set of customers C into L subsets $\mathcal{B}_l$, satisfying $\mathcal{B}_1 \cup \ldots \cup \mathcal{B}_L = C$ and $\mathcal{B}_l \cap \mathcal{B}_{l'} = \emptyset$ for all $l, l' \in L$, called the disjoint budget uncertainty sets (Gounaris et al., 2013). The budget uncertainty set (17) ensures that all possible demand realizations $\boldsymbol{d} = \{d_{ip} | d_{ip} \forall \, i \in C, p \in P\}$ must satisfy two conditions $\underline{\boldsymbol{d}} \leq \boldsymbol{d} \leq \bar{\boldsymbol{d}}$ and $\sum_{i\in\mathcal{B}_l} \boldsymbol{d_i} \leq \boldsymbol{B_l}$ for all $l \in \mathcal{L}$.

The uncertainty set $\mathcal{U}(\widetilde{\mathcal{D}})$ is defined by the intersection of a box (hyper-rectangle) uncertainty set and budget polyhedral constraints first introduced by Bertsimas and Sim (2004). More specifically, we note the following.

- The box set (i.e., deviation in demand) is defined by a scale parameter $\boldsymbol{\varepsilon} \in [0, 1]^{|P|}$ and nominal (expected) demand $\hat{\boldsymbol{d}}$ (i.e., the mean of demand). Values $\underline{\boldsymbol{d}}$ (lower demand bound) and $\bar{\boldsymbol{d}}$ (upper demand bound) are formally expressed as $\underline{\boldsymbol{d}} = \{\underline{d}_{ip} | \underline{d}_{ip} = (1 - \varepsilon_p)\hat{d}_{ip} \; \forall \, i \in C, p \in P\}$ and $\bar{\boldsymbol{d}} = \{\bar{d}_{ip} | \bar{d}_{ip} = (1 + \varepsilon_p)\hat{d}_{ip} \; \forall \, i \in C, p \in P\}$.

  Remark 1: When $\varepsilon = 0$, the demand volume becomes certain values (i.e., deterministic VRPCD - no deviation exists for any customer demand). Lower bound value $\underline{d}$ is ensured to be non-negative if $0 \leq \varepsilon \leq 1$.

- The budget polyhedral constraints imply a practical concept, in which it is unlikely that all customer demands simultaneously reach the highest demand $\bar{\boldsymbol{d}}$. To express this idea, the budget values are defined by another scalar parameter $\boldsymbol{\Gamma} \in [0, 1]^{L\times|P|}$ and the budget uncertainty sets $\mathcal{L}$ as follows.

$$\boldsymbol{B_l} = \{b_{l,p} | b_{l,p} = \sum_{i\in\mathcal{B}_l} \hat{d}_{ip} + \Gamma_{lp} \sum_{i\in\mathcal{B}_l} (\bar{d}_{ip} - \hat{d}_{ip}) \; \forall \, p \in P\} \qquad \text{for all } l \in \mathcal{L}. \tag{18}$$

  Remark 2: When $0 \leq \Gamma \leq 1$, the uncertainty set is defined by an intersection of the box with the budget constraint. When $\Gamma > 1$, the uncertainty set is only defined by the box set (see Figure 2 and the example below).

To illustrate the budget uncertainty set, we consider a network including 6 customers, 4 suppliers, and a cross-dock. We consider three subsets: $L = 3$, with $\mathcal{B}_1 = \{1, 2\}, \mathcal{B}_2 = \{3, 4\}$, and $\mathcal{B}_3 = \{5, 6\}$. Below, we describe an example (see also Figure 2) about subset $\mathcal{B}_1 = \{1, 2\}$, where for the sake of simplicity, the index of the product p is omitted.

- For the box set, inputs are provided including nominal demand $\hat{d}_1 = 3$, $\hat{d}_2 = 2$ and scalar parameters $\varepsilon_1 = \frac{2}{3}$, $\varepsilon_2 = \frac{1}{2}$. The deviations in demand by customers 1 and 2 are then defined as $[\underline{d}_1, \bar{d}_1] = [1, 5]$ and $[\underline{d}_2, \bar{d}_2] = [1, 3]$, respectively. The box set of the subset $\mathcal{B}_1$ is represented by a rectangular with four corner points consisting of $(\underline{d}_1, \underline{d}_2) = (1, 1)$, $(\underline{d}_1, \bar{d}_2) = (1, 3)$, $(\bar{d}_1, \underline{d}_2) = (5, 1)$, and $(\bar{d}_1, \bar{d}_2) = (5, 3)$.

- For the budget polyhedral set, the expression associated with subset $\mathcal{B}_1$ is $d_1 + d_2 \leq b_1$, which represents the budget constraint. We consider 4 levels of the budget set $\mathcal{B}_1$ (i.e., $\Gamma_1 = 0, 0.3, 1$, and 1.3), which correspond to four budget values $b_1 = 5, 5.9, 8$, and 8.9, respectively, evaluated by equation (18). The budget constraints are then illustrated by green lines.

- The uncertainty set, for each level of the budget set, is then defined by the intersection of the box set and the budget polyhedral set, as defined by the gray regions in the figure. For Cases 1 and 2 ($0 \leq \Gamma < 1$), the uncertainty sets are formed by the intersection of the box set and the budget constraints (Figures 2a and 2b). Moreover, the budget constraint passes through the central point $(\hat{d}_1, \hat{d}_2) = (3, 2)$ when $\Gamma = 0$ (Case 1). For Cases 3 and 4 ($\Gamma_1 \geq 1$), the uncertainty sets only depend on the box set (Figures 2c and 2d).

4.2. Robust Miller-Tucker-Zemlin Formulation

This section formulates the robust MTZ constraints to address the demand uncertainty set $\mathcal{U}(\mathcal{D})$ described in Section 4.1. Using MTZ constraints allows us to derive a compact mathematical formulation for VRPCD-DU that can be solved directly using a general-purpose mixed-integer programming solver. Moreover, based on the analysis reported in Gounaris et al. (2013), the MTZ-based formulation is one of the most effective formulations for the robust

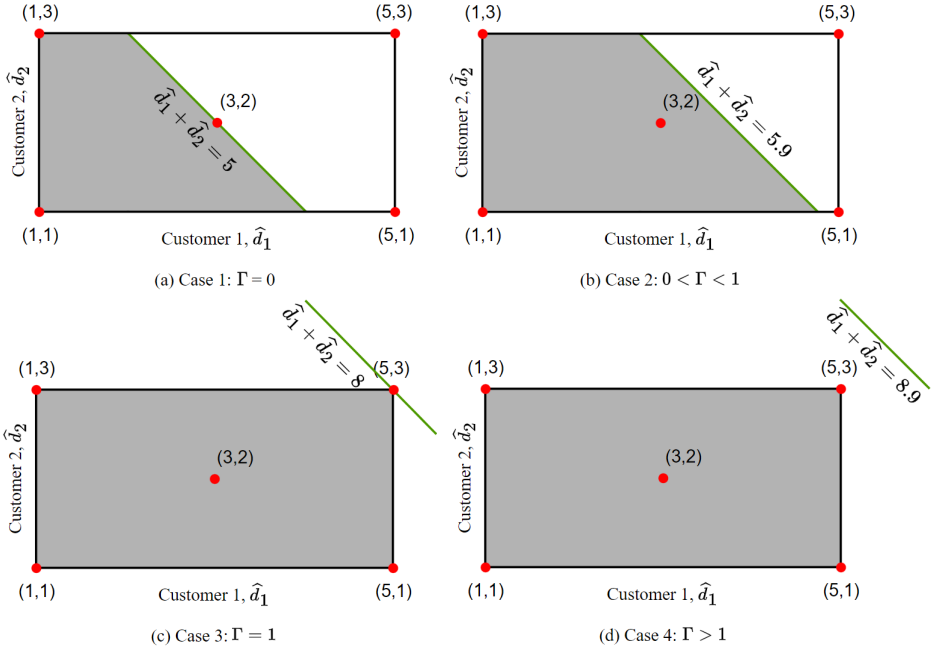


Figure 2: Graphical representation of uncertainty sets with four cases of the budget parameter Γ

CVRP. The constraints are derived from the deterministic MTZ constraints (12) to (15) by satisfying all realizations of the demand uncertainty set $\boldsymbol{d} \in \mathcal{U}(\mathcal{D})$, requiring therefore that the following constraints are satisfied:

$$u_j^{'} - u_i^{'} + Q(1 - \sum_{v \in V} x_{ijv}^{'}) \geq \sum_{h \in C} d_{hj} \qquad \forall\ (i,j) \in A^{'}, j \neq 0, \text{ and } \boldsymbol{d} \in \mathcal{U}(\mathcal{D}) \tag{19}$$

$$\sum_{h \in C} d_{hi} \leq u_i^{'} \leq Q \qquad \forall\ i \in S \text{ and } \boldsymbol{d} \in \mathcal{U}(\mathcal{D}) \tag{20}$$

$$u_j^{''} - u_i^{''} + Q(1 - \sum_{v \in V} x_{ijv}^{''}) \geq \sum_{p \in P} d_{jp} \qquad \forall\ (i,j) \in A^{''}, j \neq 0, \text{ and } \boldsymbol{d} \in \mathcal{U}(\mathcal{D}) \tag{21}$$

$$\sum_{p \in P} d_{ip} \leq u_i^{''} \leq Q \qquad \forall\ i \in C \text{ and } \boldsymbol{d} \in \mathcal{U}(\mathcal{D}) \tag{22}$$

Robust constraints (21) and (22) are derived by constraints (14) and (15), respectively. Moreover, supply quantities depend on customer demand $s_j = \sum_{i \in C} \hat{d}_{ij}$ for all $j \in S$ (see Section 3). Therefore, constraints (19) and (20) are the robust forms of constraints (12) and (13), respectively. However, the robust forms (19) to (22) currently require the solution of an infinite-dimensional problem. Below we exploit an equivalent representation in finitely decision variables (see also Gounaris et al., 2013).

First, let $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_{r'}, \mathcal{R}_{r'+1}, \ldots, \mathcal{R}_{r'+r''})$ denote a robustness solution including a set of r' routes serving the supplier nodes and a set of $r^{''}$ routes serving the customer nodes. A route $\mathcal{R}_v$ starts at the cross-dock 0, serves n_v customer/supplier nodes, and finally returns at cross-dock 0, denoted by $\mathcal{R}_v = (R_{v,0},\ R_{v,1}, \ldots,\ R_{v,n_v},\ R_{v,n_v+1})$ with $R_{v,0} = R_{v,n_v+1} = 0$ and $\{R_{v,1}, \ldots, R_{v,n_v}\} \in S$ or $\in C$. The cumulative supply $u_i^{'}$ and demand $u_i^{''}$ are defined as:

$$u_j^{'} = \sum_{i=1}^{h} \sum_{k \in C} d_{k,R_{v,i}},\ \forall j = R_{v,h}, 1 \leq h \leq n_v, v \in V \quad \text{and} \quad u_j^{''} = \sum_{i=1}^{h} \sum_{p \in P} d_{R_{v,i},p},\ \forall j = R_{v,h}, 1 \leq h \leq n_v, v \in V.$$

Decision variables $a_{ij}^{'}$ and $a_{ij}^{''}$ are then introduced to replace the robustness solution $\mathcal{R}$ in the above form. These variables indicate whether node i is visited after node j in the same route. To formalize it, let $a_{ij}^{'}$ (resp. $a_{ij}^{''}$) be 1 if

there exists a supplier (resp. customer) route $\mathscr{R}_v$ such that $i \in \mathscr{R}_v$ and $j \in \{R_{v,1}, \ldots, R_{v,h} = i\}$ and otherwise 0. We thus have:

$$u_i^{'} = \sum_{j \in S} \sum_{h \in C} a_{ij}^{'} d_{hj}, \ \forall i \in S \quad \text{and} \quad u_i^{''} = \sum_{j \in C} \sum_{p \in P} a_{ij}^{''} d_{jp}, \ \forall i \in C. \tag{23}$$

Using expressions (23) for the robust MTZ constraints (19) to (22), we obtain an equivalent expression represented by semi-infinite constraints (24) to (27).

$$\sum_{k \in S} \sum_{h \in C} \left(a_{jk}^{'} - a_{ik}^{'} \right) d_{hk} + Q(1 - \sum_{v \in V} x_{ijv}^{'}) \geq \sum_{h \in C} d_{hj} \qquad \forall \ (i,j) \in A^{'}, j \neq 0, \text{ and } \boldsymbol{d} \in \mathscr{U}(\mathscr{D}) \tag{24}$$

$$\sum_{h \in C} d_{hi} \leq \sum_{k \in S} \sum_{h \in C} a_{ik}^{'} d_{hk} \leq Q \qquad \forall \ i \in S \text{ and } \boldsymbol{d} \in \mathscr{U}(\mathscr{D}) \tag{25}$$

$$\sum_{k \in C} \sum_{p \in P} \left(a_{jk}^{''} - a_{ik}^{''} \right) d_{kp} + Q(1 - \sum_{v \in V} x_{ijv}^{''}) \geq \sum_{p \in P} d_{jp} \qquad \forall \ (i,j) \in A^{''}, j \neq 0, \text{ and } \boldsymbol{d} \in \mathscr{U}(\mathscr{D}) \tag{26}$$

$$\sum_{p \in P} d_{ip} \leq \sum_{k \in C} \sum_{p \in P} a_{ik}^{''} d_{kp} \leq Q \qquad \forall \ i \in C \text{ and } \boldsymbol{d} \in \mathscr{U}(\mathscr{D}) \tag{27}$$

Based on Gounaris et al. (2013), the original problem is considered as a two-stage robust optimization in which the routing decision is taken before the uncertain demands $\boldsymbol{d} \in \mathscr{U}(\mathscr{D})$ are realized. The routing decision is defined in the first stage denoted by $\kappa \in \mathscr{Z}$ and functions $f : \mathscr{Z} \longmapsto \mathbb{R}^{|S|}(\mathbb{R}^{C|})$ and $t : \mathscr{Z} \longmapsto \mathbb{R}$. The second stage is then expressed as semi-infinite constraints $e^{\intercal} \boldsymbol{d}^{\intercal} f(\kappa) e \leq t(\kappa) \ \forall \ \boldsymbol{d} \in \mathscr{U}(\mathscr{D})$ as the general form for constraints (24) to (27) where $e^{\intercal}$ and e are vectors of all ones. We then obtain:

$$\begin{aligned}
&\iff \max_{\boldsymbol{d} \in \mathscr{U}(\mathscr{D})} \{ e^{\intercal} \boldsymbol{d}^{\intercal} f(\kappa) e : W \boldsymbol{d} \leq \boldsymbol{B}; \boldsymbol{d} \leq \bar{\boldsymbol{d}}; \boldsymbol{d} \geq \underline{\boldsymbol{d}} \} \leq t(\kappa) \\
&\iff \min_{\Upsilon^1, \Upsilon^2, \Upsilon^3} \{ e^{\intercal} \boldsymbol{B}^{\intercal} \Upsilon^1 e + e^{\intercal} \bar{\boldsymbol{d}}^{\intercal} \Upsilon^2 e + e^{\intercal} \underline{\boldsymbol{d}}^{\intercal} \Upsilon^3 e : W^{\intercal} \Upsilon^1 + \Upsilon^2 + \Upsilon^3 \geq f(\kappa) \} \leq t(\kappa) \\
&\iff e^{\intercal} \boldsymbol{B}^{\intercal} \Upsilon^1 e + e^{\intercal} \bar{\boldsymbol{d}}^{\intercal} \Upsilon^2 e + e^{\intercal} \underline{\boldsymbol{d}}^{\intercal} \Upsilon^3 e \leq t(\kappa) \text{ and } W^{\intercal} \Upsilon^1 + \Upsilon^2 + \Upsilon^3 \geq f(\kappa)
\end{aligned} \tag{28}$$

where $W = \{ W_{li} = 1 \text{ if } i \in \mathscr{B}_l; \text{ otherwise } 0 | l \in \mathscr{L}, i \in C \}$

The first equivalence ensures the definition of $\mathscr{U}(\mathscr{D})$ in the semi-infinite constraints, which represent a general form for constraints (24) to (27). Next, the second and the third equivalences are guaranteed by the linear duality properties. The robust Miller-Tucker-Zemlin formulation is as follows. Note that each robust MTZ constraint is transformed into its dual form (28) by introducing new associated dual variables. Notation $[k = j]$ equals 1 if $k = j$ and otherwise 0.

Minimize (1)

Subject to: Constraints (2) to (11), (16), and:

$$\sum_{k \in S} \sum_{l \in \mathscr{L}} B_{lk} \varphi_{ijkl}^{'(1)} + \sum_{k \in S} \sum_{h \in C} \left(\bar{d}_{hk} \varphi_{ijhk}^{'(2)} + \underline{d}_{hk} \varphi_{ijhk}^{'(3)} \right) \leq Q(1 - \sum_{v \in V} x_{ijv}^{'}) \quad \forall \ (i,j) \in A^{'} \text{ and } j \neq 0 \tag{29}$$

$$\varphi_{ijhk}^{'(2)} + \varphi_{ijhk}^{'(3)} + \sum_{l \in \mathscr{L}} W_{lh} \varphi_{ijkl}^{'(1)} \geq a_{ik}^{'} - a_{jk}^{'} + [k = j] \qquad \forall \ (i,j) \in A^{'}, \ j \neq 0, \ h \in C, \ k \in S \tag{30}$$

Constraints (29) and (30) are the dual form of (24) with its associated dual variables consisting of $\varphi_{ijkl}^{'(1)}$ and $\varphi_{ijhk}^{'(2)}$ being non-negative variables and $\varphi_{ijhk}^{'(3)}$ being non-positive variables.

$$\sum_{k \in S} \sum_{l \in \mathscr{L}} B_{lk} \psi_{ikl}^{'(1)} + \sum_{k \in S} \sum_{h \in C} \left(\bar{d}_{hk} \psi_{ihk}^{'(2)} + \underline{d}_{hk} \psi_{ihk}^{'(3)} \right) \leq 0 \qquad \forall \ i \in S \tag{31}$$

$$\psi_{ihk}^{'(2)} + \psi_{ihk}^{'(3)} + \sum_{l \in \mathscr{L}} W_{lh} \psi_{ikl}^{'(1)} \geq [k = i] - a_{ik}^{'} \qquad \forall \ i, k \in S \text{ and } h \in C \tag{32}$$

$$\sum_{k \in S} \sum_{l \in \mathscr{L}} B_{lk} \vartheta_{ikl}^{'(1)} + \sum_{k \in S} \sum_{h \in C} \left(\bar{d}_{hk} \vartheta_{ihk}^{'(2)} + \underline{d}_{hk} \vartheta_{ihk}^{'(3)} \right) \leq Q \qquad \forall \ i \in S \tag{33}$$

$$\vartheta'^{(2)}_{ihk} + \vartheta'^{(3)}_{ihk} + \sum_{l\in\mathscr{L}} W_{lh}\vartheta'^{(1)}_{ikl} \geq a'_{ik} \qquad \forall\, i,k \in S \text{ and } h \in C \tag{34}$$

Constraints (31) to (34) are the dual form of (25) with its associated dual variables consisting of $\psi'^{(1)}_{ikl}, \psi'^{(2)}_{ihk}, \vartheta'^{(1)}_{ikl}$, and $\vartheta'^{(2)}_{ihk}$ as non-negative variables, and $\psi'^{(3)}_{ihk}$ and $\vartheta'^{(3)}_{ihk}$ are non-positive variables.

$$\sum_{p\in P}\sum_{l\in\mathscr{L}} B_{lp}\varphi''^{(1)}_{ijpl} + \sum_{p\in P}\sum_{k\in C}\left(\bar{d}_{kp}\varphi''^{(2)}_{ijkp} + \underline{d}_{kp}\varphi''^{(3)}_{ijkp}\right) \leq Q(1 - \sum_{v\in V} x''_{ijv}) \quad \forall\ (i,j) \in A'' \text{ and } j \neq 0 \tag{35}$$

$$\varphi''^{(2)}_{ijkp} + \varphi''^{(3)}_{ijkp} + \sum_{l\in\mathscr{L}} W_{lk}\varphi''^{(1)}_{ijpl} \geq a''_{ik} - a''_{jk} + [k=j] \qquad \forall\,(i,j)\in A'',\ j\neq 0,\ k\in C,\ p\in P \tag{36}$$

$$\sum_{p\in P}\sum_{l\in\mathscr{L}} B_{lp}\psi''^{(1)}_{ipl} + \sum_{p\in S}\sum_{k\in C}\left(\bar{d}_{kp}\psi''^{(2)}_{ikp} + \underline{d}_{kp}\psi''^{(3)}_{ikp}\right) \leq 0 \qquad \forall\, i \in C \tag{37}$$

$$\psi''^{(2)}_{ikp} + \psi''^{(3)}_{ikp} + \sum_{l\in\mathscr{L}} W_{lk}\psi''^{(1)}_{ipl} \geq [k=i] - a''_{ik} \qquad \forall\, i,k \in C \text{ and } p \in P \tag{38}$$

$$\sum_{p\in P}\sum_{l\in\mathscr{L}} B_{lp}\vartheta''^{(1)}_{ipl} + \sum_{p\in P}\sum_{k\in C}\left(\bar{d}_{kp}\vartheta''^{(2)}_{ikp} + \underline{d}_{kp}\vartheta''^{(3)}_{ikp}\right) \leq Q \qquad \forall\, i \in C \tag{39}$$

$$\vartheta''^{(2)}_{ikp} + \vartheta''^{(3)}_{ikp} + \sum_{l\in\mathscr{L}} W_{lk}\vartheta''^{(1)}_{ipl} \geq a''_{ik} \qquad \forall\, i,k \in C \text{ and } p \in P \tag{40}$$

Dual forms of the robust MTZ constraints (26) and (27) are similarly presented by (35) to (40) with dual variables consisting of $\varphi''^{(1)}_{ijpl}, \varphi''^{(2)}_{ijhp}, \psi''^{(1)}_{ipl}, \psi''^{(2)}_{ihp}, \vartheta''^{(1)}_{ipl}$, and $\vartheta''^{(2)}_{ihp}$ as non-negative variables. Moreover, $\varphi''^{(3)}_{ijhp}, \psi''^{(3)}_{ihp}$, and $\vartheta''^{(3)}_{ihp}$ are non-positive variables.

4.3. Handling the robust solution under demand uncertainty

This section discusses an efficient implementation of the procedure used to check the feasibility of a solution in terms of the robust capacity constraint. In particular, we discuss how to compute the cumulative load of a vehicle route under demand uncertainty and how the cumulative load is updated under the insertion and removal of customers.

4.3.1. Computing the cumulative loading

The MTZ constraints cannot be easily used in the context of designing effective heuristic algorithms, where efficient feasibility checks must be frequently executed to verify solutions. It is essential to optimize their computation in the context of a heuristic algorithm where several insertion and removal operations are executed. We therefore adopt robust alternative constraints based on the rounded capacity inequalities (RCI) proposed by Gounaris et al. (2013) for CVRP. As shown by the recent work of Wang et al. (2022), robust capacity constraints are also used in their state-of-the-art branch-price-and-cut robust approach for the robust CVRP. The constraints are defined as follows:

$$\sum_{i\in S^+\setminus S^-}\sum_{j\in S^-} x'_{ij} \geq \left\lceil \frac{1}{Q}\max_{\boldsymbol{d}\in\mathscr{U}(\mathscr{D})}\sum_{k\in S^-}\sum_{h\in C} d_{hk}\right\rceil, \qquad \forall\, S^- \subseteq S, \tag{41}$$

$$\sum_{i\in C^+\setminus C^-}\sum_{j\in C^-} x''_{ij} \geq \left\lceil \frac{1}{Q}\max_{\boldsymbol{d}\in\mathscr{U}(\mathscr{D})}\sum_{k\in C^-}\sum_{p\in P} d_{kp}\right\rceil, \qquad \forall\, C^- \subseteq C. \tag{42}$$

The right-hand side (RHS) of inequalities (41) and (42) compute lower bounds on the minimum number of vehicles required to serve the set of nodes S^- and C^-, respectively. Given the set of routes composing a solution, we aim to verify whether the solution is feasible for the robust constraints - that is, to verify if the following inequalities are satisfied:

$$\max_{\boldsymbol{d}\in\mathscr{U}(\mathscr{D})}\sum_{k\in\mathscr{R}_v}\sum_{h\in C} d_{hk} \leq Q, \qquad \forall\, v = 1,\ldots,r', \tag{43}$$

$$\max_{\boldsymbol{d}\in\mathscr{U}(\mathscr{D})}\sum_{k\in\mathscr{R}_v}\sum_{p\in P} d_{kp} \leq Q, \qquad \forall\, v = r'+1,\ldots,r'+r''. \tag{44}$$

Constraints (43) and (44) ensure that the robust loading of supplier and customer routes does not exceed the vehicle capacity Q, respectively. Let $\mathcal{Q}_{(\mathcal{R}_v)}$, for $v = 1, \dots, r' + r''$, denote the maximum values of the left-hand side (LHS) of inequalities (43) and (44). Below we discuss in detail the computation of values $\mathcal{Q}_{(\mathcal{R}_v)}$.

Computing $\mathcal{Q}_{(\mathcal{R}_v)}$ for the supplier routes

We observe for a given supplier route $\mathcal{R}_v$ and a supplier $k \in \mathcal{R}_v$ visited by the route that the maximum value $\bar{s}_k$ of the demand associated with k over set $\mathcal{U}(\mathcal{D})$ is a constant and does not depend on the route composition. Indeed, the maximum value can be computed by:

$$\bar{s}_k = \max \sum_{h \in C} d_{hk} = \sum_{h \in C} \underline{d}_{hk} + \sum_{l \in \mathcal{L}} \sum_{h \in \mathcal{B}_l} \left(b_{lk} - \underline{d}_{hk} \right), \tag{45}$$

and therefore the values $\mathcal{Q}_{(\mathcal{R}_v)}$ can be easily computed as:

$$\mathcal{Q}_{(\mathcal{R}_v)} = \sum_{k \in \mathcal{R}_v} \bar{s}_k, \qquad \forall\, v = 1, \dots r'. \tag{46}$$

Computing $\mathcal{Q}_{(\mathcal{R}_v)}$ for the customer routes

In the LHS of (44), for $v = r' + 1, \dots, r''$, $\mathcal{Q}_{(\mathcal{R}_v)}$ can be computed in polynomial time based on the method proposed by Gounaris et al. (2013) (see Proposition 10 of their paper) and the fact that $\bigcup_{l=1}^{L} \mathcal{B}_l = C$ as follows:

$$\mathcal{Q}_{(\mathcal{R}_v)} = \sum_{k \in \mathcal{R}_v} \sum_{p \in P} \underline{d}_{kp} + \sum_{p \in P} \sum_{l \in \mathcal{L}} \min \left(b_{lp} - \sum_{i \in \mathcal{B}_l} \underline{d}_{ip}, \sum_{i \in \mathcal{B}_l \cap \mathcal{R}_v} \left(\bar{d}_{ip} - \underline{d}_{ip} \right) \right), \qquad \forall\, v = r' + 1, \dots, r''. \tag{47}$$

The complexity term for computing function (47) for a given route $\mathcal{R}_v$ is therefore $\mathcal{O}(|\mathcal{R}_v| \times |P|)$. We observe that if the uncertainty set is defined by a box set, then expression (47) reduces to $\mathcal{Q}_{(\mathcal{R}_v)} = \sum_{k \in \mathcal{R}_v} \sum_{p \in P} \bar{d}_{kp}$.

We next show that term $\mathcal{Q}_{(\mathcal{R}_v)}$ can also be computed by solving a maximum flow problem based on the Ford-Fulkerson algorithm (Ray, 2013). Moreover, by handling maximum flow solutions, we also present that the complexity of computing (47) when a route is updated by inserting or removing a customer can be reduced to $\mathcal{O}(|P|)$.

For sake of the exposition, in the following description we omit the use of the index p. Let $\mathcal{R}_v = \{0, i_1, i_2, \dots, i_h, 0\}$ be a customer route visiting customers $C^- = \{i_1, i_2, \dots, i_h\}$. To illustrate the maximum flow problem, we use a widely adopted notation in graph theory (see, for example, Ray, 2013). We are given a capacitated network $\mathcal{G} = (\mathcal{V}, \mathcal{A})$,as shown in Figure 3, where the set of vertices $\mathcal{V}$ is defined as $\{s, t\} \cup C^- \cup \{\mathcal{B}_l : l \in \mathcal{L}, \mathcal{B}_l \cap C^- \neq \emptyset\}$ (i.e., source, sink, visited customer, and its corresponding budget vertices), and the set of arcs $\mathcal{A}$ with corresponding arc capacities $c(i, j)$, $(i, j) \in \mathcal{A}$, is defined as follows.

- $\{(s, i) : i \in C^-\}$, where each arc (s, i) is associated with capacity $c(s, i) = \bar{d}_i$ (see the blue arcs in Figure 3).
- $\{(i, t) : i \in C^-\}$, where each arc (i, t) is associated with capacity $c(i, t) = \underline{d}_i$ (red arcs).
- $\{(i, \mathcal{B}_l) : l \in \mathcal{L}, i \in \mathcal{B}_l \cap C^-\}$, where each arc $(i, \mathcal{B}_l)$ is associated with capacity $c(i, \mathcal{B}_l) = \delta d_i = \bar{d}_i - \underline{d}_i$ (green arcs).
- $\{(\mathcal{B}_l, t) : l \in \mathcal{L}, \mathcal{B}_l \cap C^- \neq \emptyset\}$, where each arc $(\mathcal{B}_l, t)$ is associated with capacity $c(\mathcal{B}_l, t) = \beta_l = \left(b_l - \sum_{i \in \mathcal{B}_l} \underline{d}_i \right)$ (black arcs).

We assume that $\hat{\boldsymbol{d}} < \bar{\boldsymbol{d}}$ where $\hat{\boldsymbol{d}} = \max\{\boldsymbol{d} \in \mathcal{U}(\mathcal{D})\}$; i.e., under the uncertainty set $\mathcal{U}(\mathcal{D})$ each customer cannot achieve its maximum demand in $\bar{\boldsymbol{d}}$. Under this assumption, let $(T, \bar{T})$ be an s-t cut of graph $\mathcal{G}$ where $\{s, C^-\} \subseteq T$ and $t \in \bar{T}$. Let $\mathcal{L}' \subseteq \mathcal{L}$ be the set of vertices included in set T. We note that the capacity of the s-t cut $(T, \bar{T})$ is equal to:

$$\sum_{i \in C^-} \underline{d}_i + \sum_{l \in \mathcal{L}'} \left(b_l - \sum_{i \in \mathcal{B}_l} \underline{d}_i \right) + \sum_{l \in \mathcal{L} \setminus \mathcal{L}'} \sum_{i \in \mathcal{B}_l \cap C^-} \left(\bar{d}_i - \underline{d}_i \right). \tag{48}$$

It is easy to see that the above expression corresponds to the computation of the RHS of equation (47) where the terms of the minimization argument are selected by set $\mathcal{L}'$ and $\mathcal{L} \setminus \mathcal{L}'$. Hence, the capacity of the minimum s-t cut of graph $\mathcal{G}$ corresponds to $\mathcal{Q}_{(\mathcal{R}_v)}$. We clarify it by providing an illustrative example, which Appendix A discusses.

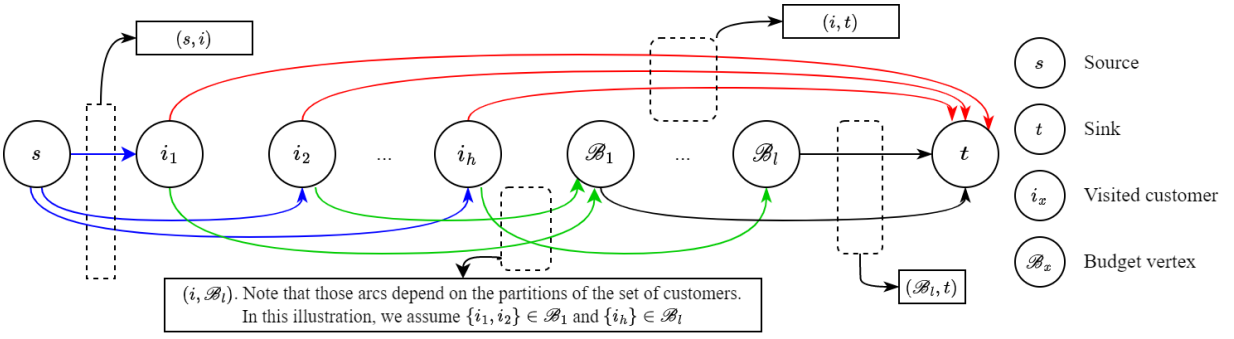


Figure 3: Schematic representation of the capacitated graph $\mathscr{G} = (\mathscr{V}, \mathscr{A})$

4.3.2. Updating vehicle routes

We describe below how $\mathscr{Q}_{(\mathscr{R}_v)}$ can be updated when a customer is removed from route $\mathscr{R}_v$ or a new customer is inserted into the route. The following additional coefficients are used.

- $\rho_l, l \in \mathscr{L}$: the total capacity of the arcs entering nodes $\mathscr{B}_l$; i.e., $\sum_{j \in \mathscr{B}_l \cap C^-} c(j, \mathscr{B}_l)$; equal to the sum of differences between upper and lower bounds of customers in the subset $\mathscr{B}_l \cap C^-$, computed as $\sum_{i \in \mathscr{B}_l \cap C^-} \left(\bar{d}_i - \underline{d}_i\right)$.
- $\pi_l, l \in \mathscr{L}$: the flow of arc $(\mathscr{B}_l, t)$, which represents the actual uncertain quantity of budget l associated with route $\mathscr{R}_v$.

The above coefficients are initialized to zero whenever a new route is created. The terms “before" and “after" are used below to denote old and new values of coefficients, respectively.

Updating the load based on a customer removal

Let i be the customer to be removed from route $\mathscr{R}_v$ where $i \in \mathscr{B}_l$. Let customers h and k be the predecessor and successor of customer i, respectively; i.e., we have $\mathscr{R}_v = \{\dots, h, i, k, \dots\}$. When customer i is removed from the route, value $\mathscr{Q}_{(\mathscr{R}_v)}$ must be updated. This can be done by means of an update of the different coefficients in order as follows:

$$\rho_l^{after} = \rho_l^{before} - \delta d_i. \tag{49}$$

$$\pi_l^{after} = \min\left(\rho_l^{after}, \beta_l\right). \tag{50}$$

$$\mathscr{Q}_{(\mathscr{R}_v)}^{after} = \mathscr{Q}_{(\mathscr{R}_v)}^{before} - \underline{d}_i - \left[\pi_l^{before} - \pi_l^{after}\right]^+. \tag{51}$$

Updating the load based on a customer insertion

Let i be the customer to be inserted into route $\mathscr{R}_v$ where $i \in \mathscr{B}_l$. Suppose that customer i is added between customers h and k; i.e., $\mathscr{R}_v = \{\dots, h, i, k, \dots\}$. The coefficients are updated as follows:

$$\rho_l^{after} = \rho_l^{before} + \delta d_i. \tag{52}$$

$$\pi_l^{after} = \min\left(\rho_l^{after}, \beta_l\right). \tag{53}$$

$$\mathscr{Q}_{(\mathscr{R}_v)}^{after} = \mathscr{Q}_{(\mathscr{R}_v)}^{before} + \underline{d}_i + \left[\pi_l^{after} - \pi_l^{before}\right]^+. \tag{54}$$

To summarize, demand uncertainty is handed in the proposed ALNS by using Equations (49) - (54). Appendix A provides an example of computation based on the above expressions.

5. An ALNS algorithm for VRPCD-DU

Our algorithm is based on the ALNS framework, first introduced by Ropke and Pisinger (2006). The algorithm starts by generating an initial solution inspired by the greedy saving heuristic (see Section 5.1). At each algorithm

iteration, a new neighborhood solution is explored by using destroy and repair methods, and the algorithm terminates if no improvements are found after a predefined number θ of iterations. For each iteration, the probability for selecting a destroy or repair method is determined by the performance of operators during the search (see Section 5.4).

Algorithm 1 shows the details of ALNS. Let sol denote the incumbent (current) solution at the beginning of each iteration, $sol^{'}$ denote a temporary solution created by the destroy and repair operators, and sol^{*} denote the global best solution. The algorithm initializes the temperature T at the initial value T_0, which is reduced by α after every η_{Cool} iterations (Van Breedam, 1995). Let NI denote a variable to count the number of consecutive times that no new global best solution is found after every η_{ALNS} iterations. The weights, scores, and probabilities of operators are then also updated. The algorithm terminates when $NI = \theta$.

Regarding the ALNS operators, the degree of destruction/reconstruction is also important and is controlled by η, which denotes the number of nodes removed from and re-inserted into a solution. If η is too small, then the solution is hard to escape the local optimal, but if η is too large, then it is difficult to reconstruct a good solution and also takes a long computational time (Sacramento et al., 2019). Thus, at each iteration, η is randomly determined in $\eta \in [\eta_{min}, \eta_{max}]$, and it will be tuned for obtaining the best value of those parameters (see Section 6.2).

Regarding the acceptance criteria, there are three cases for accepting the solution $sol^{'}$ after finishing operators are the new incumbent solution sol and even the new global best solution sol^{*}. First, $sol^{'}$ becomes the global best solution sol^{*} if $sol^{'}$ is feasible and better than sol^{*} (see lines 18-22). Second, $sol^{'}$ replaces the incumbent solution sol for the next iteration if $sol^{'}$ is better than sol (lines 24-26). Third, the worse solution can be accepted to be the new incumbent solution with a probability of $e^{-\Delta/T}$. Let Δ denote the difference between the total cost of sol and the total cost of sol' given by $\Delta = TC(sol) - TC(sol')$ (lines 28-30). For the two latter cases, feasibility is ignored since the search space allows infeasible solutions with some rules when implementing operators (discussed in Section 5.6).

A solution representation sol is constructed by a two-dimensional array consisting of a fixed number of $|V|$ rows representing the vehicles routes (i.e., supplier's routes $sol^{(s)}$ and customers routes $sol^{(c)}$). Every vehicle starts from the cross-dock 0, visits several nodes, and returns to cross-dock 0. Through the solution representation, the flow conservation of routes is always ensured. A feasibility check therefore considers the maximum duration constraint (16) and the vehicle capacity constraints (43) and (44). This is executed whenever a destroy or repair operator is considered. In particular, the solution will be first updated (see Section 4.3) and then feasibility is checked.

5.1. Initial solution

Algorithm 2 shows the steps used to generate an initial solution. The algorithm starts initializing the set of unassigned customers U_C and suppliers U_S by the set of customers C and suppliers S, respectively. Empty (unused) routes are then defined by starting and ending at cross-dock 0. Let $route$ be the route index and $index$ be the position for adding a node into $route$. Moreover, when adding a node to the solution, it is costly to the objective value, calculated by $loss$. In the algorithm, customers and suppliers are sequentially assigned until those sets are empty (lines 4 - 17). For each process, the feasible insertion of a customer node (Cus) or a supplier node (Sup) is found by the greedy savings heuristic (Algorithm 3), in which its output contains the route index rd, the position for inserting id, and the loss of insertion $temp$. The best insertion is chosen to minimize $loss$, which is stored by the inserted node $node$, the route index $route$, and the position for inserting $index$. Thus, $node$ is eliminated from unassigned sets, and solutions are updated when $node$ is added into the solution (see Section 4.3).

Algorithm 3 shows the greedy savings heuristic in more detail. The input includes a node and the set of routes $sol_0^{(c)}$ or $sol_0^{(s)}$, which we generally call $sol_0^{(x)}$. To ensure that the initial solution is feasible, we just consider to insert nodes into an existing route if it maintains the feasibility of solutions (lines 7-9). If no feasible insertion exists, then the insertion into a new route is proposed (line 2).

$$loss = \begin{cases} c'_{i,node} + c'_{node,j} - c'_{ij} & \text{if inserting into an existing route and } node \in S \\ c'_{i,node} + c'_{node,j} + F & \text{if inserting into a new route and } node \in S \\ c''_{i,node} + c''_{node,j} - c''_{ij} & \text{if inserting into an existing route and } node \in C \\ c''_{i,node} + c''_{node,j} + F & \text{if inserting into a new route and } node \in C \end{cases} \tag{55}$$

Algorithm 1: ALNS pseudocode

Input: an initial solution sol_0, set of destroy/repair operators Ω^+ and Ω^-, $T \leftarrow T_0$
Output: sol^*

Define sol^*, sol', $sol \leftarrow sol_0$
Initialize weights ω_j, scores π_j, probabilities λ_i of destroy/repair operators // Section 5.4
Set $iter, NI \leftarrow 0$
$foundbest \leftarrow$ FALSE
repeat
  if $(iter \mod \eta_{ALNS}) = 0$ **then**
    Update weights ω_i and probabilities λ_i of operators based on π_i // Section 5.4
    Reset the scores of operators π_i to 0
    if $foundbest = TRUE$ **then**
      $NI \leftarrow 0$
    else
      $NI \leftarrow NI + 1$
    $foundbest \leftarrow$ FALSE
  Select D_x, I_y from Ω^+, Ω^- based on its probabilities λ_i (Roulette wheel selection)
  $\eta \leftarrow$ Generate from $U \sim [\eta_{min}, \eta_{max}]$
  Destroy sol', $RemovedSet \leftarrow D_x(sol, \eta)$ // Section 5.2
  Repair $sol' \leftarrow I_y(sol', RemovedSet)$ // Section 5.3
  if sol' *is better than* sol^* **then**
    if sol' *is feasible* **then**
      Update sol, $sol^* \leftarrow sol'$
      $foundbest \leftarrow$ TRUE
      Add ζ_1 value to π_x and π_y // ζ_1 is mentioned in Section 5.4
  else
    if sol' *is better than* sol **then**
      Update $sol \leftarrow sol'$
      Add ζ_2 value to π_x and π_y // ζ_2 is mentioned in Section 5.4
    else
      if *random(0,1)* $< \exp^{\Delta/T}$ **then**
        Update $sol \leftarrow sol'$
        Add ζ_3 value to π_x and π_y // ζ_3 is mentioned in Section 5.4
  if $(iter \mod \eta_{Cool}) = 0$ **then**
    $T \leftarrow \alpha T$
  $iter \leftarrow iter + 1$
until $NI \geq \theta$

5.2. Destroy operators

This section discusses seven removals. Each operator is denoted by $D_x(sol, \eta)$, where the input includes a complete solution sol and a predefined number η. The operator then modifies solution sol by removing some customers, represented by set J_C, and some suppliers, represented by set J_S.

- **Random removal heuristic** $D_1(sol, \eta)$: This operator randomly removes η nodes (supplier/customer) from the solution. This approach can help in diversifying the search space (Ropke and Pisinger, 2006). To be aware of the peculiarities of our problem (i.e., two separated processes), at each time we select a removed node based on the probabilities of selecting customer $p_C = \frac{|C|}{|C|+|S|}$ and supplier $p_S = 1 - p_C$.

Algorithm 2: Initial solution for VRPCD-DU

Input: set of unassigned customers $U_C \leftarrow C$ and suppliers $U_S \leftarrow S$
Output: $sol_0 \leftarrow sol_0^{(c)} \cup sol_0^{(s)}$

Set $sol_0^{(c)}, sol_0^{(s)} \leftarrow \{[0,0]\}$
Set $index, route \leftarrow 0$, $loss \leftarrow BigM$
repeat
  if $U_C \neq \emptyset$ **then**
    for Cus *in* U_C **do**
      $id,\ rd,\ temp \leftarrow Greedy(Cus, sol_0^{(c)})$
      if $loss > temp$ **then**
        Update $index, route, node \leftarrow id, rd, Cus$
    Update $U_C \leftarrow U_C \setminus \{node\}$
    Update $sol_0, sol_0^{(c)} \leftarrow$ add $node$ into route $route$ at position $index$ // Section 4.3
  if $U_S \neq \emptyset$ **then**
    for Sup *in* U_S **do**
      $id,\ rd,\ temp \leftarrow Greedy(Sup, sol_0^{(s)})$
      if $loss > temp$ **then**
        Update $index, route, node \leftarrow id, rd, Sup$
    Update $U_S \leftarrow U_S \setminus \{node\}$
    Update $sol_0, sol_0^{(s)} \leftarrow$ add $node$ into route $route$ at position $index$ // Section 4.3
until $U_C \neq \emptyset$ *and* $U_S \neq \emptyset$

Algorithm 3: Greedy Savings Heuristic

Input: $node, sol_0^{(x)}$
Output: $id, rd, loss$

Evaluate loss by equation (55) for inserting into a new route
$newRoute \leftarrow True$ // TRUE: a new route is created
for r *in* $sol_0^{(x)}$ **do**
  for pos *in* $route$ **do**
    Evaluate $temp$ to insert $node$ into route r at position pos // Equation (55)
    if $temp < loss$ **then**
      if $\mathcal{Q}_{(r)} \leq Q$ *and* $TT \leq T_{max}$ **then** // mentioned in Section 4.3
        Update $id, rd, loss \leftarrow pos, r, temp$
        $newRoute \leftarrow False$

- **Shaw removal heuristic** $D_2(sol, \eta)$: This removal heuristic was first proposed by Shaw (1997). This removal aims to remove η nodes that are similar in terms of travel cost, travel time, and node position. We denote ϕ_1, ϕ_2, and ϕ_3 as the weights of mentioned factors, respectively. Let l_{ij} be 0, if nodes i and j are in the same vehicle and otherwise 1. A lower $Related(i,j)$ value means the more related nodes i and j are. Parameter p_{worst} controls randomness in the selection (a low value corresponds to much randomness). The procedure of this removal is thus illustrated by Algorithm 4.

$$Related(i,j) = \begin{cases} \phi_1\left(c'_{i,j} + c'_{j,i}\right) + \phi_2\left(t'_{ij} + t'_{ji}\right) + \phi_3 e_{ij}, & \text{if } i \in S \\ \phi_1\left(c''_{i,j} + c''_{j,i}\right) + \phi_2\left(t''_{ij} + t''_{ji}\right) + \phi_3 e_{ij}, & \text{if } i \in C \end{cases} \tag{56}$$

Algorithm 4: Shaw removal heuristic

Input: sol, η
Output: sol, J_C, J_S
repeat
  if $random(0,1) < p_c$ **then**
    if $J_c = \emptyset$ **then**
      $i \leftarrow$Select a randomly customer node
      Update sol by removing node i from $sol^{(c)}$
      $J_C \leftarrow J_C \cup \{i\}$
    else
      $i \leftarrow$ Select a random node in J_C
      Set $Z = \{j | j \in C \setminus J_C\}$, stored by ascending $Related(i,j)$ // Evaluate by Eq. (56)
      $j \leftarrow Z\left[\lceil \xi p_{worst} \rceil\right]$ // $\xi \sim U(0,1)$
      Update sol by removing node j from $sol^{(c)}$
      $J_C \leftarrow J_C \cup \{j\}$
  else
    if $J_S = \emptyset$ **then**
      $i \leftarrow$ Select a randomly supplier node
      Update sol by removing node i from $sol^{(s)}$
      $J_S \leftarrow J_S \cup \{i\}$
    else
      $i \leftarrow$ Select a random node in J_S
      Set $Z = \{j | j \in S \setminus J_S\}$, stored by ascending $Related(i,j)$ // Evaluate by Eq. (56)
      $j \leftarrow Z\left[\lceil \xi p_{worst} \rceil\right]$ // $\xi \sim U(0,1)$
      Update sol by removing node j from $sol^{(s)}$
      $J_S \leftarrow J_S \cup \{j\}$
until $|J_C| + |J_S| = \eta$

- **Neighborhood removal heuristics** $D_3(sol, \eta)$ and $D_4(sol, \eta)$: Both the two removal operators are special cases of D_2, which Demir et al. (2012) named as demand-based and time-based removals, respectively. The underlying difference is the weights of related factors, where D_3 just mentions the travel cost with $\left(\phi_1, \phi_2, \phi_3\right) = (1, 0, 0)$, and D_4 just considers the travel time with $\left(\phi_1, \phi_2, \phi_3\right) = (0, 1, 0)$.

- **Worst-cost node removal heuristic** $D_5(sol, \eta)$: This operator iteratively removes nodes with the highest difference between the total cost of the solution before and after removing, denoted by $gain_i$ (Demir et al., 2012). The selection randomness of this operator is also controlled by p_{worst}. The framework is shown in Algorithm 5.

$$gain_i = \begin{cases} c'_{pre(i),i} + c'_{i,suc(i)} - c'_{pre(i),suc(i)}, & \text{if } i \in S \quad \text{using for } D_5 \\ c''_{pre(i),i} + c''_{i,suc(i)} - c''_{pre(i),suc(i)}, & \text{if } i \in C \quad \text{using for } D_5 \\ t'_{pre(i),i} + t'_{i,suc(i)} - t'_{pre(i),suc(i)}, & \text{if } i \in S \quad \text{using for } D_6 \\ t''_{pre(i),i} + t''_{i,suc(i)} - t''_{pre(i),suc(i)}, & \text{if } i \in C \quad \text{using for } D_6 \end{cases} \tag{57}$$

- **Worst-time node removal heuristic** $D_6(sol, \eta)$: This operator is similar to operator D_5, but it considers travel time instead of travel cost (Demir et al., 2012).

Algorithm 5: Worst node removal heuristics

Input: sol, η
Output: sol, J_C, J_S
repeat
  Set $Z = \{i | i \in (C \setminus J_C) \cup (S \setminus J_S)\}$, stored by descending $gain_i$ // Evaluate by Eq. (57)
  $i \leftarrow Z\left[\lceil \xi p_{worst} \rceil\right]$ // $\xi \sim U(0,1)$
  if $i \in C$ **then**
    Update sol by removing node i from $sol^{(c)}$
    $J_C \leftarrow J_C \cup \{i\}$
  else
    Update sol by removing node i from $sol^{(s)}$
    $J_S \leftarrow J_S \cup \{i\}$
until $|J_C| + |J_S| = \eta$

- **Worst route removal heuristic** $D_7(sol, \eta)$: The aim of this operator is to moderate the time resource of each process by removing some or all nodes in the bottleneck routes. It creates more potential insertions to reconstruct the feasible solution later, and it is implemented by selecting the routes with the largest travel time. Algorithm 6 illustrates the operator.

Algorithm 6: Worst route removal heuristic

Input: sol, η
Output: sol, J_C, J_S
repeat
  if $random(0,1) < 0.5$ **then**
    $route \leftarrow \max_{r \in sol^{(c)}} t_r$ // t_r is mentioned in Section 4.3
    $z \leftarrow \min\left(\eta - |J_C| - |J_S|, n_{route}\right)$
    repeat
      $i \leftarrow$The first node in route $route$
      Update sol by removing node i from $sol^{(c)}$
      $J_C \leftarrow J_C \cup \{i\}$
    until z *nodes are removed*
  else
    $route \leftarrow \max_{r \in sol^{(s)}} t_r$ // t_r is mentioned in Section 4.3
    $z \leftarrow \min\left(\eta - |J_C| - |J_S|, n_{route}\right)$
    repeat
      $i \leftarrow$The first node in route $route$
      Update sol by removing node i from $sol^{(s)}$
      $J_S \leftarrow J_S \cup \{i\}$
    until z *nodes are removed*
until $|J_C| + |J_S| = \eta$

5.3. Repair operators

After deconstructing the solution, we repair the solution using nine insertion operators. A repair operator is denoted by $I_y(sol, J_C, J_S)$, where the input includes an incomplete solution sol and sets of removed customer nodes J_C and supplier nodes J_S. All nodes in sets J_C and J_S are then inserted into solution sol to obtain a complete solution.

- **Greedy insertion heuristic with travel cost** $I_1(sol, J_C, J_S)$: This operator was first proposed by Ropke and Pisinger (2006). It aims at inserting a node to a position with the lowest insertion cost. Let $cost_j$ denote the cost

of inserting node j in its best position; i.e., the minimum cost position of node j. We then choose the removed node that minimizes the minimum cost insertion. The framework of I_1 is shown in Algorithm 7.

Algorithm 7: Greedy insertion heuristic

Input: sol, J_C, J_S
Output: sol
repeat
  if $random(0,1) < p_c$ **then**
    for *each node j in* J_C **do**
      $id, rd, cost_j \leftarrow Greedy(j, sol^{(c)})$ // the minimum cost position of node j
      if $loss > cost_j$ **then**
        Update $index, route, node \leftarrow id, rd, j$
      Update sol by removing node from $sol^{(c)}$
      $J_C \leftarrow J_C \setminus \{node\}$
  else
    for *each node j in* J_S **do**
      $id, rd, cost_j \leftarrow Greedy(j, sol^{(s)})$ // the minimum cost position of node j
      if $loss > cost_j$ **then**
        Update $index, route, node \leftarrow id, rd, j$
      Update sol by removing node from $sol^{(s)}$
      $J_S \leftarrow J_S \setminus \{node\}$
until $|J_C| \cup |J_S| = \emptyset$

- **Greedy insertion heuristic with travel time** $I_2(sol, J_C, J_S)$: Based on operator I_1, the travel time is considered instead of the travel cost. For intuition, when the time resource for each insertion is minimized, more potential insertions could remain for the next nodes. We hence expect to obtain better solutions.
- ***k*−regret insertion heuristic with travel cost** $I_3(sol, J_C, J_S)$ and $I_4(sol, J_C, J_S)$: these operators are based on a similar operator proposed by Ropke and Pisinger (2006), aimed at improving greedy operators like I_1 and I_2. We consider 3−regret and 4−regret for I_3 and I_4, respectively. The node with the maximum regret is selected. The framework of those heuristics is similar to Algorithm 7.
- ***k*−regret insertion heuristics with travel time** $I_5(sol, J_C, J_S)$ and $I_6(sol, J_C, J_S)$: These operators are similar to operators I_3 and I_4, respectively, where travel time is considered instead of travel cost.
- **Least routes travel time heuristic** $I_7(sol, J_C, J_S)$: This operator focuses on the utilization of routes in terms of travel time resources, where vehicles with lower travel times have higher priorities to insert unassigned nodes. For the implementation, all routes are first sorted in ascending order with respect to the route's travel time. Instead of observing all routes to find the best insertion as in operator I_1, each route is sequentially selected in the order for finding the insertion. This operator will stop searching when nodes can be assigned to the considered route. This operator explores the search space and may take less computation time than I_1 and I_2.
- **Greedy insertion heuristics with noise functions** $I_8(sol, J_C, J_S)$ and $I_9(sol, J_C, J_S)$: These operators are based on the basic greedy insertion operators I_1 and I_2, respectively, with a degree of randomness (Demir et al., 2012). The selection randomness is additionally considered by a noise function $noise = \xi y_{noise}$, where y_{noise} is a tuned parameter and $\xi \sim U(-1, 1)$.

$$loss = \begin{cases} c'_{i,node} + c'_{node,j} - c'_{ij} + \xi y_{noise} & \text{if inserting into an existing route and } node \in S \\ c'_{i,node} + c'_{node,j} + F + \xi y_{noise} & \text{if inserting into a new route and } node \in S \\ c''_{i,node} + c''_{node} - c''_{ij} + \xi y_{noise} & \text{if inserting into an existing route and } node \in C \\ c''_{i,node} + c''_{node,j} + F + \xi y_{noise} & \text{if inserting into a new route and } node \in C \end{cases} \tag{58}$$

5.4. Adaptive weight adjustment

Let Ω^+ and Ω^- denote the set of destroy and repair operators, respectively. To select the removal and insertion heuristics more efficiently, we consider a weight ω_i for each operator, which represents how well the operator i has performed. Based on weights ω_i, the probability for selecting an operator i (λ_i) is evaluated by (59).

$$\lambda_i = \frac{\omega_i}{\sum_{j=1\dots|\Omega^o|} \omega_j} \qquad \forall\, i = 1 \dots |\Omega^o|, o = \{-,+\} \tag{59}$$

The weight of each operator is initialized to ω_0 (a user-defined parameter). The weight of each operator is then adjusted based on how well it has performed in the last *segment*, where each segment t includes η_{ALNS} iterations. We propose the score π_i to keep track of the performance of operator or heuristic i, whereby a higher score corresponds to an efficient heuristic. Coefficient π_i is set to 0 at the start of each segment. Next, π_i is increased by one of three ζ_1, ζ_2, and ζ_3 values as shown in Table C4 when the heuristic i is used as shown in Algorithm 1. Let θ_i be the number of times we have used operator i during the segment t. The coefficients w_i are updated as follows:

$$\omega_{i,t+1} = (1-\gamma)\,\omega_{i,t} + \gamma\frac{\pi_i}{\theta_i} \qquad \forall\, i = 1 \dots |\Omega^-| \text{ and } i = 1 \dots |\Omega^+|. \tag{60}$$

5.5. Updating travel time and total cost

Travel times and costs are not affected by the uncertainty parameters. To update the vehicles total travel times and costs, we use methods similar to the methods proposed by Gunawan et al. (2021), which work as follows.

- Let TC be the total cost of the solution. We only need to add and subtract arc costs relating to the modified position. In addition, if a new route is created (resp. removed), then the total cost should be added to (resp. subtracted from) the operational cost F in constant time. This can be done in constant time.
- Let TT be the total time to complete both supplier and customer processes, computed by $TT = T_{SP} + T_{CD}$; and $t_v^{'}$ and $t_v^{''}$ denote the travel time of route v in supplier and customer processes, respectively. When an operator is implemented in the supplier (resp. customer) vehicle v, we first update the travel time $t_v^{'}$ (resp. $t_v^{''}$). We then update $T_{SP} = \max(t_v^{''}, T_{SP})$ (resp. $T_{CD} = \max(t_v^{''}, T_{CD})$). The solution is checked for whether TT exceeds T_{max} or not. This can be done in constant time.

5.6. Search space

The literature shows that ALNS algorithms exploring both the feasible and infeasible solution spaces may lead to improved solution methods (Adulyasak et al., 2014; Dayarian et al., 2016; Hemmelmayr et al., 2012; Ribeiro and Laporte, 2012).

To our best knowledge, state-of-the-art algorithms for solving VRPCD (Lee et al., 2006) have not yet considered searching the infeasible solution space. Therefore, in our algorithm we allow the violation of the constraint on the total duration T_{max} based on the following rules.

- During the execution of a destroy operator, the constraint on T_{max} is not checked for feasibility.
- During the execution of a repair operator, the constraint on T_{max} is not checked for feasibility, but an insertion is accepted only if the resulting violation in duration is not greater than the violation of the original solution.
- An improved solution is accepted as the new best solution if it is also feasible for the maximum duration constraint.

6. Computational experiments

In this section we present the results of extensive computational experiments. Section 6.1 describes the test instances used in our experiments. Section 6.2 illustrates how the different ALNS parameters are calibrated. Section 6.3 gives the results on VRPCD together with a comparison versus the state-of-the-art algorithms. Sections 6.5 and 6.6 report the results on VRPCD-DU and insights on the robustness of the solutions, respectively.

All experiments were performed on a computer equipped with an Intelő Core i7-10700 CPU @ 2.9GHz and 32GB of RAM. The commercial solver GUROBI is used to solve the MILP models associated with the robust formulation.

6.1. Test instances

Our benchmark set is based on the set of VRPCD instances generated by Lee et al. (2006) and extended by Gunawan et al. (2020) to the case of multiple products. The instances are available at the website https://www.mech.kuleuven.be/en/cib/op/opmainpage#section-53.

The instances are grouped into three subsets: 10 nodes with 10 vehicles (Set 1), 30 nodes with 20 vehicles (Set 2), and 50 nodes with 20 vehicles (Set 3) with 30 instances in each subset. The details of the number of suppliers $|S|$, number of customers $|C|$, fleet size $|V|$, vehicle capacity Q, operation cost F, planning time horizon T_{max}, travel distances, travel times, and nominal customer demand are shown in the supplementary materials (Table S.1). In particular, travel distances, travel times, and demands are drawn from uniform distributions.

In our experiments we consider the two parameter values $\boldsymbol{\varepsilon}$ and $\boldsymbol{\Gamma}$ introduced in Section 3 to define the budget uncertainty set $\mathscr{U}(\mathscr{D})$. More precisely, parameter $\boldsymbol{\varepsilon}$ controls the maximal demand deviation corresponding to the nominal demand, where $\pm\boldsymbol{\varepsilon}\hat{\boldsymbol{d}}$ represents the lower and upper bounds of demands; parameter $\boldsymbol{\Gamma}$ models L uncertainty budgets involving customer subsets $\mathscr{B}_l$, where $\bigcup_{l=1}^{L}\mathscr{B}_l = C$ with exactly $L = 3,\ 4,\ 4$ for Sets 1, 2, and 3, respectively. We assume that the maximum number of customers in each subset is defined by $h_{max} = \left\lceil \frac{|C|}{L} \right\rceil$. To partition the set of customers C into L subsets, we use the following two different approaches.

(1) "Random" method (P_1): We simply partition the set of customers by the index of customers. Next, we have $\mathscr{B}_l = \{\left(h_{max}(l-1)+i\right) \in C |\ \forall\, i = 1 \dots h_{max}\},\ \forall\, l = 1 \dots L$. More specifically, for Set 1: $\mathscr{B}_1 = \{1, 2\},\ \mathscr{B}_2 = \{3, 4\},\ \mathscr{B}_3 = \{5, 6\}$; for Set 2: $\mathscr{B}_1 = \{1, \dots, 6\},\ \mathscr{B}_2 = \{7, \dots, 12\},\ \mathscr{B}_3 = \{13, \dots, 18\},\ \mathscr{B}_4 = \{19, \dots, 23\}$; and for Set 3: $\mathscr{B}_1 = \{1, \dots, 10\},\ \mathscr{B}_2 = \{11, \dots, 20\},\ \mathscr{B}_3 = \{11, \dots, 29\},\ \mathscr{B}_4 = \{30, \dots, 38\}$.

(2) "Clustered" method (P_2): We partition the set of customers based on the Capacitated p-Median Problem (CPMP) (Maniezzo et al., 1998), in which the dissimilarity between two customers is stipulated by the cost matrix. We use the following CPMP model to partition the set of customers into L disjoint clusters:

$$\textit{(CPMP)} \qquad \text{Minimize} \sum_{i\in C}\sum_{j\in C} x_{ij}(c''_{ij} + c''_{ji})$$

$$\sum_{i\in C} x_{ij} \le h_{max} y_j \qquad \forall\, j \in C$$

$$\sum_{j\in C} x_{ij} = 1 \qquad \forall\, i \in C$$

$$\sum_{j\in C} y_j = L$$

$$x_{ij} \in \{0,1\}\ \forall\, i, j \in C \text{ and } y_j \in \{0,1\}\ \forall\, j \in C,$$

where $x_{ij} = 1$ if customer i belongs to the subset (cluster) where customer j is the median and 0 otherwise; and $y_j = 1$ if customer j is the median of a subset (cluster). Note that we consider all customers to be potential medians. The details of the resulting partitions are given in the supplementary materials (Tables S.2 to S.4).

In reality we can link the two aforementioned parameters to their practical meanings. Parameter ε relates to the range of actual demand, which depends on the type of the product (e.g., dairy products, household products, fresh food). Parameter Γ simulates the fact that customer demands are unlikely to reach their maximum values at the same time, but the cumulative demand of each customer subset is limited by budget constraints (i.e., a box set). These subsets are partitioned by customer attributes, such as geographic location or other attributes. Decision makers estimate Γ, whose higher value leads to a higher conservative solution and vice versa.

We vary the values of the two parameters for the uncertainty level $(\boldsymbol{\varepsilon}, \boldsymbol{\Gamma})$ to extensively analyze the effects of demand uncertainty on the routing plans. Let $A = \{0.2k | k \in \{1, \dots, 3\}\}$ and $B = \{0.25k | k \in \{0, \dots, 4\}\}$ denote the ranges of value for $\boldsymbol{\varepsilon}$ and $\boldsymbol{\Gamma}$, respectively. We combine 3 possible values of $\boldsymbol{\varepsilon}$ and 5 possible values of $\boldsymbol{\Gamma}$ to obtain $3 \times 5 = 15$ combinations of $(\boldsymbol{\varepsilon}, \boldsymbol{\Gamma})$ that represent the uncertainty sets $\mathscr{U}(\mathscr{D})$ as shown in Table C5.

6.2. Parameter settings

A careful tuning of the parameters may improve the efficiency and effectiveness of metaheuristics (Ridge and Kudenko, 2007). Our ALNS algorithm uses 17 parameters whose values must be determined. To this end, we select 10 medium- and large-scale instances as a test bed. The *one factor at a time* (OFAT) approach is adopted for parameter

Table 3
Summary comparison of the proposed ALNS versus BKS and other algorithms.

	TS		imp-TS		SA		Matheuristic (BKS)		ALNS	
	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best
Set 1	19.61	-	5.24	-	1.57	-	0.00	0.00	0.00	0.00
Set 2	87.17	-	19.66	-	17.13	-	0.00	0.00	-1.73	-0.34
Set 3	61.67	-	37.58	-	29.32	-	0.00	0.00	-2.34	-1.49

calibration (see Yu et al., 2021). We first set each parameter to its initial value; all parameters are listed in Table C6, where the order reported in the table is the order in which the parameters have been evaluated. We sequentially consider parameters of the acceptance and stopping criteria, ALNS criteria, and destroy/repair operators. We test all candidate values for each parameter, and the remaining values are fixed by the initial values unless the best values were defined. For a candidate value, we test 20 selected instances and record the best solutions of 5 runs. We then determine the best-tuned value leading to the lowest average results of all tested instances. This process is repeated until all best-tuned values are found. Table C6 summarizes the initial, all candidates', and final values of the parameters.

6.3. Results on VRPCD

In this section we report the results obtained by the ALNS algorithm on VRPCD. The results are compared with the results of the state-of-art algorithms including TS (Lee et al., 2006), imp-TS (Liao et al., 2010), SA (Yu et al., 2016), and matheuristic (Gunawan et al., 2021).

For each algorithm and a given instance, the gap from the optimal or best-known solution is computed as $GAP(\%) = \frac{TC^* - TC_{Algorithm}}{TC^*}$, where TC^* is the cost of the best-known solution, and $TC_{Algorithm}$ is the cost of the solution computed by the algorithm. For Set 1 instances, values TC^* are optimal values computed by Yu et al. (2016) using a MIP model. For Set 2 and 3 instances, the best-known values are taken from Gunawan et al. (2021).

Detailed results obtained on the three set of instances (i.e., Sets 1, 2, and 3) are presented in Tables C7 to C9, respectively. For each instance, the tables report the cost of the best solutions known, and for each algorithm the average cost of the best solutions is obtained over 5 runs (Avg.) and the average CPU times in seconds (CPU). For the ALNS algorithm, the cost of the best solution found is also reported. The last two sections of the tables show the percentage gaps of the different algorithms based on the average solution costs and the best solution costs, respectively.

Table 3 summarizes the results on VRPCD instances and reports a comparison with the state-of-the-art algorithms in terms of solution quality, represented by the gaps between the results from each algorithm and BKS. For Set 1 instances, our algorithm provides the optimal solutions for all instances in all 5 runs. For Set 2 and 3 instances, the proposed ALNS outperforms all the other algorithms. More precisely, the BKS average solutions for both Set 2 and 3 instances are improved by 1.73% and 2.34% on average, respectively. The best results after 5 runs are also compared with the best results of the Gunawan et al. (2021) approach. As from the results, our algorithm finds 18 new best solutions. The ALNS algorithm also obtains 21 new best solutions.

Regarding the computational time, Table C10 lists the performances of the machines used by the different approaches based on single-thread CPU benchmarks computed by PassMark Software, which is generally used to fairly compare different machines. Based on the benchmarks, we normalize the computing time of each algorithm as $\overline{CPU}_{Algorithm} = \frac{rate_{Algorithm}}{rate_{ALNS}} CPU_{Algorithm}$. The average normalized CPU time of our algorithm is shorter than SA and matheuristic algorithms and longer than TS and imp-TS algorithms as shown in Table 4. However, the solution quality of TS and imp-TS is much worse than our algorithm. Our proposed ALNS therefore outperforms the state-of-the-art algorithms to solve VRPCD instances.

It is worth noting that the Set 1, 2, and 3 VRPCD instances are characterized by tight maximum duration constraints. Indeed, the average of the time utilization of the solutions, where the utilization is computed as the ratio between the total time of the vehicle routes and parameter T_{max}, equals 85.9%, 99.4%, and 99.4% for Set 1, 2, and 3 instances, respectively.

6.4. Analysis of the search space in ALNS

This section analyzes the effectiveness of considering infeasible solutions during the search (see Section 5.6). We consider the following results.

Table 4
Normalize the average computation time of algorithms with respect to ALNS (Unit: seconds).

	Average actual computing time (CPU)					Average normalized computing time ($\overline{CPU}$)				
	TS	imp-TS	SA	Math	ALNS	TS	imp-TS	SA	Math	ALNS
Set 1	2.02	0.12	2.06	0.16	0.06	0.38	0.02	1.62	0.15	0.06
Set 2	2.86	0.26	3.02	1.7	1.13	0.54	0.05	2.37	1.55	1.13
Set 3	7.82	0.41	7.19	6.41	3.42	1.47	0.08	5.65	5.85	3.42

Table 5
Summary of the comparison of the results from ALNS-f versus BKS and the proposed ALNS.

	Set	Average Solution				Best solution			
		Worse	Better	Equal	GAP (%)	Worse	Better	Equal	GAP (%)
BKS	Set 1	0	0	30	0.00	0	0	30	0.00
	Set 2	0	30	0	-1.64	10	15	5	-0.16
	Set 3	2	28	0	-2.15	12	18	0	-0.81
ALNS	Set 1	0	0	30	0.00	0	0	30	0.00
	Set 2	18	12	0	0.10	16	11	3	0.19
	Set 3	18	12	0	0.21	19	11	0	0.70

- ALNS: the results from the proposed ALNS are obtained in Section 6.3, where the algorithm allows infeasible solutions during the search.
- ALNS-f: ALNS is modified by forbidding infeasible solutions in the search space, named by ALNS-f. The results from ALNS-f are provided by solving VRPCD instances over 5 runs.
- BKS: BKS solutions of VRPCD instances are also mentioned in Section 6.3.

Table 5 summarizes the comparison of the results from ALNS-f to BKS and to the results from ALNS. For Set 1 instances, ALNS-f also obtains the optimal solutions for all instances. For Set 2 and 3 instances, ALNS-f provides better solutions than BKS, but worse solutions than ALNS. Regarding ALNS-f, we note the following.

- To BKS, the average (resp. best) solutions are improved 1.64% and 2.15% (resp. 0.16% and 0.81%) on average for Sets 2 and 3, respectively. In Set 2, 28 (resp. 18) out of 30 average (resp. best) solutions are better. In Set 3, 30 (resp. 15) out of 30 average (resp. best) solutions are better.
- To ALNS, the average (resp. best) solutions are worse than by 0.10% and 0.21% (resp. 0.19% and 0.7%) on average for Sets 2 and 3, respectively. Although ALNS-f provides worse solutions on average, there are still 11 (resp. 12) out of average (resp. best) solutions that are better than ALNS solutions.

The difference between the average CPU of ALNS and ALNS-f is not significant, which is ignored in this discussion. Therefore, the proposed ALNS with an infeasible solution space during the search efficiently improves the quality of solutions in the VRPCD instances.

6.5. Results on VRPCD-DU

We first consider the VRPCD-DU instances with uncertainty level $(\boldsymbol{\varepsilon}, \boldsymbol{\Gamma}) = (0.6, 0.5)$ and the "Clustered" partition method (P_2), which are solved using both the GUROBI MIP solver and the ALNS algorithm. The results on the three sets of instances are given in Table C11, taking a notation similar to the previous tables. We impose a time limit of 14,400 and 28,800 seconds to GUROBI to solve Set 2 and 3 instances, respectively.

For Set 1, all the instances are solved to optimality by GUROBI and also by the ALNS algorithm with a significant speed up of the solution time. For Sets 2 and 3, the GUROBI solver is not able to compute optimal solutions within the imposed time limit and computes heuristic solutions only for instances of Set 2. On Set 2 instances, the ALNS algorithm always computes improved solutions, with an average improvement equal to about 20% within a limited computing time, thanks to the efficiency of the procedure used to check the feasibility of a solution (see also Section

Table 6
Average results of the robust solution with various combinations

Combination		0	1	2	3	4	5	6	7
Cost	Set 1	7529.63	7589.43	7817.27	8190.63	8308.20	8564.07	7634.40	8198.40
	Set 2	6497.47	6497.47	6497.47	6497.47	6531.93	6556.63	6497.47	6497.47
	Set 3	13826.17	13826.17	13906.10	13976.63	14004.50	14019.37	13832.10	13969.10
Ratio "CR"	Set 1	1.000	1.008	1.038	1.088	1.103	1.137	1.014	1.089
	Set 2	1.000	1.000	1.000	1.000	1.005	1.009	1.000	1.000
	Set 3	1.000	1.000	1.006	1.011	1.013	1.014	1.000	1.010
Combination		8	9	10	11	12	13	14	15
Cost	Set 1	8575.80	9212.90	9887.57	7650.20	8330.30	9357.33	10366.53	11039.13
	Set 2	6556.63	7163.63	7370.73	6497.47	6531.93	7160.87	7370.83	7368.93
	Set 3	13999.83	14076.70	14241.30	13902.40	14025.13	14141.47	14311.17	14405.37
Ratio "CR"	Set 1	1.139	1.224	1.313	1.016	1.106	1.243	1.377	1.466
	Set 2	1.009	1.103	1.134	1.000	1.005	1.102	1.134	1.134
	Set 3	1.013	1.018	1.030	1.006	1.014	1.023	1.035	1.042

4.3). The results on these instances also show that, in terms of computing times, the ALNS algorithm has similar performance in solving VRPCD and VRPCD-DU instances.

We also test the ALNS algorithm for VRPCD-DU on 15 additional uncertainty combinations $(\boldsymbol{\varepsilon}, \boldsymbol{\Gamma})$. Tables C12 to C14 show the best objective value for each instance in Sets 1, 2, and 3, again after 5 runs. The combination 0 refers to the deterministic case, and we denote the cost ratio CR with the term $\frac{TC_{VRPCD-DU}}{TC_{VRPCD}}$; i.e., the ratio between the VRPCD-DU and VRPCD solutions' costs. We also denote with unmet quantity the total demand left unserved by the vehicles when the worst-case uncertainty occurs.

Based on the two terms CR and Unmet quantity, decision markers can figure out the trade-offs between uncertain level (unmet quantity) and total cost (benefit), called the trade-off in RO (Sungur et al., 2008). Table 6 summarizes the average ratio "CR" for each set, which shows that a higher uncertainty level incurs a higher total cost (higher ratio). Moreover, we observe that the average "CR" ratio in Set 1 is significantly high (up to 1.466 in Combination 15), which means handling the demand uncertainty has a significant impact on the total cost. However, CR ratios are not significant (i.e., ≈ 0) for Sets 2 and 3, which will be further discussed in Section 6.6.

We clearly observe the impacts of the uncertain demands on the solution cost when compared with the deterministic case. As an example, Table C15 shows the cost ratios for instance 4 of Set 1.

One example about instance "10-4" of Set 1 with the VRPCD routing plan presents the impact of RO, which is analyzed under an uncertainty level (0.6, 0.5). Table C16 shows that the loading of the vehicles in the VRPCD solutions exceeds the vehicle capacity under the worst-case realizations, leading to unmet quantity. To be immune from demand uncertainty, an extra cost of about 2.5% must be paid in the VRPCD-DU routing plan (see Table C15). In addition, we provide an example showing how the uncertainty factors impact the solution structures and related costs (see Appendix B).

6.6. Analysis of the VRPCD solutions under demand uncertainty

To evaluate the effect of vehicle capacity under demand uncertainty, we first compute the utilization of the vehicles in terms of the capacity constraints for the VRPCD solutions where the demands are given by the worst-case demand realizations of the different levels of uncertainty $(\boldsymbol{\varepsilon}, \boldsymbol{\Gamma})$. Table C17 shows the results obtained in terms of Average, Min, and Max utilizations, where a utilization greater than one means that the corresponding deterministic solution is infeasible under the given demand realization. The table shows that, for Set 2 and Set 3 instances, a few solution results are infeasible, in particular involving only supplier routes.

We therefore analyze alternative vehicle capacity values, as shown by Table C18, to evaluate the impact of varying the vehicle capacity and to define suitable values for the analyses on the price of the robustness given in the next section. Based on the results obtained, we decide to use $Q = 90$ for Set 2 and $Q = 105$ for Set 3.

6.7. The price of robustness

To investigate the price of robustness, we consider VRPCD-DU instances with vehicle capacity Q equal to 70, 90, and 105 for Sets 1, 2 and 3, respectively. The following combinations of instances and partition methods are Set 1 - Cluster, Set 2 - Cluster, Set 3 - Cluster, Set 1 - Random, Set 2 - Random, and Set 3 - Random.

We use Monte Carlo simulation (Munari et al., 2019) to evaluate the performance of the RO approach under demand uncertainty. We relate the quality of VRPCD and VRPCD-DU solutions in terms of the price of robustness (PoR) and the probability of loading constraint violations (Risk).

- **The price of robustness**: PoR is evaluated by $\frac{TC(sol(\boldsymbol{\varepsilon},\boldsymbol{\Gamma}))-TC(sol)}{TC(sol)}$, where $TC(sol)$ is the solution cost of VRPCD with nominal demand, and $TC(sol(\boldsymbol{\varepsilon},\boldsymbol{\Gamma}))$ is the solution cost for a given budget of uncertainty $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma})$. It is the price for protecting against demand uncertainty.

- **The probability of constraint violation**: Values $\mathscr{P}[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}]$, $\mathscr{P}_C[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}]$, and $\mathscr{P}_S[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}]$ denote the probability of constraint violation of all the routes of solution sol, of the routes visiting the customers, and of the routes visiting the suppliers, respectively, under the uncertainty level $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma})$ and after 10,000 empirical simulations computed as shown by Algorithm 8, where sol is the solution cost of VRPCD (out-of-sample testing).

Algorithm 8: Monte Carlo simulation

Input: sol and $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma})$ for all $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma}) \in$ set of combinations
Output: $\mathscr{P}[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}]$, $\mathscr{P}_C[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}]$, $\mathscr{P}_S[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}]$

$\mathscr{F} \leftarrow 0$
for $k = 1 \dots N$ **do**
  $d_{ij} \sim \mathscr{U}(\mathscr{D})$ with $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma})$ for all $i \in C$ and $j \in S$
  if *sol violates the loading constraints* **then**
    $\mathscr{F} \leftarrow \mathscr{F} + 1$
    if *the loading violation is incurred by vehicle serving customers* **then**
      $\mathscr{F}_C \leftarrow \mathscr{F}_C + 1$
    if *the loading violation is incurred by vehicle serving suppliers* **then**
      $\mathscr{F}_S \leftarrow \mathscr{F}_S + 1$
$\mathscr{P}[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}] \leftarrow \mathscr{F}/N$; $\mathscr{P}_C[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}] \leftarrow \mathscr{F}_C/N$; $\mathscr{P}_S[sol,\boldsymbol{\varepsilon},\boldsymbol{\Gamma}] \leftarrow \mathscr{F}_S/N$

The results of the simulation are summarized by Figure 4 and the corresponding details given in Table C19. The results obtained can be summarized as follows.

- Considering PoR and $\mathscr{P}$ values with respect to the uncertainty levels, an increasing trend is observed for all six sets. We observe that Γ has more influences than the ε factor. In other words, the higher the Γ value is, the higher are the PoR and $\mathscr{P}$ values. In addition, by fixing parameter Γ, the higher the ε value is, the higher are the PoR and $\mathscr{P}$ values.

- Considering two different methods for defining the budget subsets, we observe that the PoR values for all three "cluster" sets are slightly lower than the corresponding values for the "random" sets. Therefore, the curse of conservative costs under demand uncertainty can be slightly reduced when the budget of demand uncertainty depends on "clusters".

- Considering the size of the problem, the higher is the number of nodes (i.e., suppliers and customers), the smaller is the PoR value. Typically, the PoR values of Set 3 instances are very low on average (i.e., 1.60% and 1.66% for "cluster" and "random" methods, respectively).

- Considering the values of $\mathscr{P}_C$ and $\mathscr{P}_S$, Set 1 shows similar values, whereas for Sets 2 and 3 the violation is mainly due to the loading of suppliers. In addition, the probability of loading violations of "Random instances is higher than the "Cluster instances, which means that locations of customers influence the loading of vehicles.

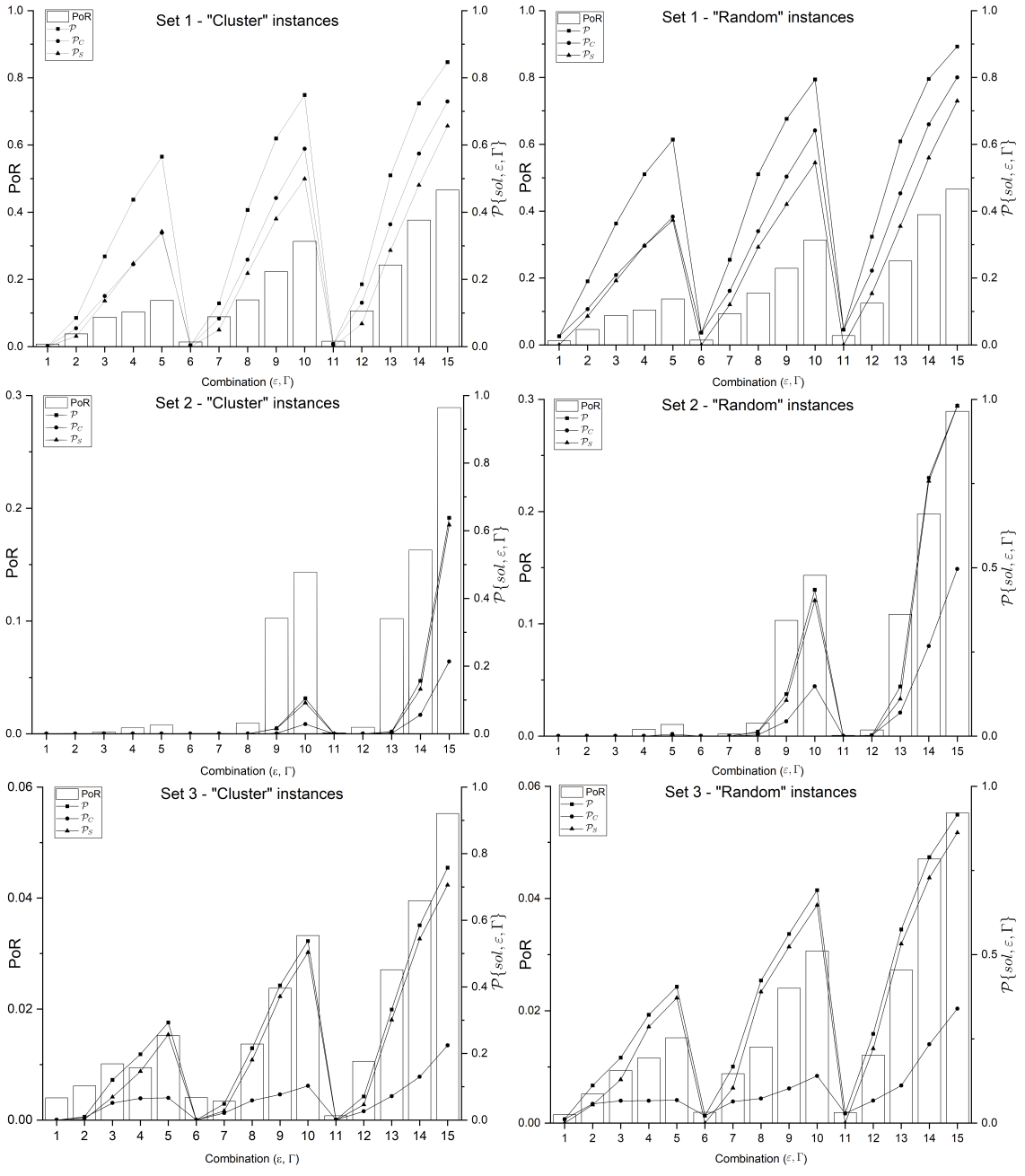


Figure 4: Average PoR, $\mathscr{P}$, $\mathscr{P}_C$, and $\mathscr{P}_S$ for each set via 15 combinations of uncertainty levels

- The uncertainty levels with $\Gamma = 0.25$ (i.e., combinations 2, 7, and 12) provide a good trade-off between PoR and $\mathscr{P}$.

- We observe that for Set 2 the values of PoR, $\mathscr{P}$, $\mathscr{P}_C$, and $\mathscr{P}_S$ are very low for combinations 1 to 8. The reason is that the utilization of the vehicle capacity does not exceed 1 under demand uncertainty in combinations 1 to 8 (see Tables C17 to C18).

7. Conclusions

We investigate a robust optimization approach to address the Vehicle Routing Problem with Cross-Docking under Demand Uncertainty (VRPCD-DU). We design a MILP model based on Miller-Tucker-Zemlin constraints for VRPCD, and its robust counterpart is derived for VRPCD-DU where uncertainty is modeled using a budget uncertainty set. We propose an ALNS algorithm for both VRPCD and VRPCD-DU relying on a new search space and new destruction and repair operators. For the VRPCD problem, our algorithm outperforms state-of-the-art algorithms. The algorithm is also extensively tested on newly generated VRPCD-DU instances.

For VRPCD-DU, we also investigate the price of robustness (PoR) and explore some managerial insights. The results show that different customers' distribution defining the budget uncertainty set shares similar PoR, and that a higher probability of capacity constraints' violation is generally observed for the suppliers. Therefore, a decision maker should carefully define the characteristics of the vehicle fleet with a focus on supplier operations, in order to be protected against uncertainty in a stronger way.

We highlight the following limitations of our work. Regarding modelling uncertain demand, we have adopted a widely used approach proposed in the literature for vehicle routing problems; i.e., robust optimization based on polyhedral uncertainty sets. Although robust optimization is a powerful technique in dealing with uncertainty in optimization, its solutions can be too conservative. Other options are available, such as stochastic optimization, as a natural choice when dealing with uncertainty, but they might suffer from two main shortcomings: information availability and tractability. Other alternative methods are also available, leading to less conservative solutions, such as adjustable robust optimization (Yanıkoğlu et al., 2019), but at the price of less computationally tractable optimization models than the optimization models derived from robust optimization. Regarding the mathematical formulation, the limitations are the size of the instances that can be solved optimally. However, in our paper the results on small-size instances are used to attest to the quality of the heuristic approach, which is designed to handle larger instances.

Finally, real-world vehicle routing problems with cross-docking pose several challenges, such as synchronization constraints at the cross-dock and uncertain travel times. The models and algorithms proposed in this paper to handle uncertainty demands can be extended to deal with other cross-docking constraints. Our future work will consider these extensions and other important features of this class of problems.

Acknowledgements

The authors would like to thank the anonymous reviewers and editor for their helpful suggestions and very thorough review of the paper.

References

Adulyasak, Y., Cordeau, J.F., Jans, R., 2014. Optimization-based adaptive large neighborhood search for the production routing problem. Transportation Science 48, 20–45.

Adulyasak, Y., Jaillet, P., 2016. Models and algorithms for stochastic and robust vehicle routing with deadlines. Transportation Science 50, 608–626.

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L.M., Poss, M., Requejo, C., 2012. Layered formulation for the robust vehicle routing problem with time windows, in: International Symposium on Combinatorial Optimization, Springer. pp. 249–260.

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L.M., Poss, M., Requejo, C., 2013. The robust vehicle routing problem with time windows. Computers & Operations Research 40, 856–866.

Bartolini, E., Goeke, D., Schneider, M., Ye, M., 2021. The robust traveling salesman problem with time windows under knapsack-constrained travel time uncertainty. Transportation Science 55, 371–394.

Ben-Tal, A., Nemirovski, A., 2002. Robust optimization–methodology and applications. Mathematical Programming 92, 453–480.

Bertsimas, D., Brown, D.B., Caramanis, C., 2011. Theory and applications of robust optimization. SIAM Review 53, 464–501.

Bertsimas, D., Sim, M., 2003. Robust discrete optimization and network flows. Mathematical Programming 98, 49–71.

Bertsimas, D., Sim, M., 2004. The price of robustness. Operations Research 52, 35–53.

Birge, J.R., Louveaux, F., 2011. Introduction to Stochastic Programming. Springer Science & Business Media.

Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I., 2016. The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99, 300–313.

Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Management Science 6, 80–91.

Dayarian, I., Crainic, T.G., Gendreau, M., Rei, W., 2016. An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem. Transportation Research Part E: Logistics and Transportation Review 95, 95–123.

De La Vega, J., Munari, P., Morabito, R., 2019. Robust optimization for the vehicle routing problem with multiple deliverymen. Central European Journal of Operations Research 27, 905–936.

Demir, E., Bektaş, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. European journal of operational research 223, 346–359.

Eufinger, L., Kurtz, J., Buchheim, C., Clausen, U., 2020. A robust approach to the capacitated vehicle routing problem with uncertain costs. INFORMS Journal on Optimization 2, 79–95.

Gounaris, C.E., Repoussis, P.P., Tarantilis, C.D., Wiesemann, W., Floudas, C.A., 2016. An adaptive memory programming framework for the robust capacitated vehicle routing problem. Transportation Science 50, 1239–1260.

Gounaris, C.E., Wiesemann, W., Floudas, C.A., 2013. The robust capacitated vehicle routing problem under demand uncertainty. Operations Research 61, 677–693.

Grangier, P., Gendreau, M., Lehuédé, F., Rousseau, L.M., 2016. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. European Journal of Operational Research 254, 80–91.

Gunawan, A., Widjaja, A.T., Gan, B., Yu, V.F., Jodiawan, P., 2020. Vehicle routing problem for multi-product cross-docking. Proceedings of the International Conference on Industrial Engineering and Operations Management 2020: Dubai, UAE .

Gunawan, A., Widjaja, A.T., Vansteenwegen, P., Yu, V.F., 2021. A matheuristic algorithm for the vehicle routing problem with cross-docking. Applied Soft Computing 103, 107163.

Hemmelmayr, V.C., Cordeau, J.F., Crainic, T.G., 2012. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Computers & Operations Research 39, 3215–3228.

Hu, C., Lu, J., Liu, X., Zhang, G., 2018. Robust vehicle routing problem with hard time windows under demand and travel time uncertainty. Computers & Operations Research 94, 139–153.

Jaillet, P., Qi, J., Sim, M., 2016. Routing optimization under uncertainty. Operations Research 64, 186–200.

Kiani Mavi, R., Goh, M., Kiani Mavi, N., Jie, F., Brown, K., Biermann, S., A Khanfar, A., 2020. Cross-docking: A systematic literature review. Sustainability 12, 4789.

Lee, C., Lee, K., Park, S., 2012. Robust vehicle routing problem with deadlines and travel time/demand uncertainty. Journal of the Operational Research Society 63, 1294–1306.

Lee, Y.H., Jung, J.W., Lee, K.M., 2006. Vehicle routing scheduling for cross-docking in the supply chain. Computers & Industrial Engineering 51, 247–256.

Liao, C.J., Lin, Y., Shih, S.C., 2010. Vehicle routing with cross-docking in the supply chain. Expert Systems with Applications 37, 6868–6873.

Lu, D., Gzara, F., 2019. The robust vehicle routing problem with time windows: Solution by branch and price and cut. European Journal of Operational Research 275, 925–938.

Luce, R.D., Raiffa, H., 1954. A Survey of the Theory of Games. Technical Report. Columbia Univ New York Bureau of Applied Social Research.

Maniezzo, V., Mingozzi, A., Baldacci, R., 1998. A bionomic approach to the capacitated p-median problem. Journal of Heuristics 4, 263–280.

Masson, R., Lehuédé, F., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transportation Science 47, 344–355.

Morais, V.W.C., Mateus, G.R., Noronha, T.F., 2014. Iterated local search heuristics for the vehicle routing problem with cross-docking. Expert Systems with Applications 41, 7495–7506.

Mousavi, S.M., Tavakkoli-Moghaddam, R., Jolai, F., 2013. A possibilistic programming approach for the location problem of multiple cross-docks and vehicle routing scheduling under uncertainty. Engineering Optimization 45, 1223–1249.

Mousavi, S.M., Vahdani, B., 2017. A robust approach to multiple vehicle location-routing problems with time windows for optimization of cross-docking under uncertainty. Journal of Intelligent & Fuzzy Systems 32, 49–62.

Mousavi, S.M., Vahdani, B., Tavakkoli-Moghaddam, R., Hashemi, H., 2014. Location of cross-docking centers and vehicle routing scheduling under uncertainty: a fuzzy possibilistic–stochastic programming model. Applied Mathematical Modelling 38, 2249–2264.

Munari, P., Moreno, A., De La Vega, J., Alem, D., Gondzio, J., Morabito, R., 2019. The robust vehicle routing problem with time windows: compact formulation and branch-price-and-cut method. Transportation Science 53, 1043–1066.

Ordóñez, F., 2010. Robust vehicle routing, in: Risk and Optimization in an Uncertain World. INFORMS, pp. 153–178.

Pisinger, D., Ropke, S., 2019. Large Neighborhood Search, in: Gendreau, M., Potvin, J.Y. (Eds.), Handbook of Metaheuristics. Springer. International Series in Operations Research & Management Science. chapter 0, pp. 99–127. URL: https://ideas.repec.org/h/spr/isochp/978-3-319-91086-4_4.html, doi:10.1007/978-3-319-91086-4.

Rahbari, A., Nasiri, M.M., 2017. Robust vehicle routing and cross-dock scheduling with uncertain loading and unloading time, in: 1st International Conference on Systems Optimization & Business Management.

Rahbari, A., Nasiri, M.M., Werner, F., Musavi, M., Jolai, F., 2019. The vehicle routing and scheduling problem with cross-docking for perishable products under uncertainty: Two robust bi-objective models. Applied Mathematical Modelling 70, 605–625.

Ray, S.S., 2013. Graph theory with algorithms and its applications: in applied science and technology. Springer.

Ribeiro, G.M., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Computers & Operations Research 39, 728–735.

Ridge, E., Kudenko, D., 2007. Tuning the performance of the mmas heuristic, in: International Workshop on Engineering Stochastic Local Search Algorithms, Springer. pp. 46–60.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40, 455–472.

Sacramento, D., Pisinger, D., Ropke, S., 2019. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. Transportation Research Part C: Emerging Technologies 102, 289–315.

Santos, M.J., Curcio, E., Mulati, M.H., Amorim, P., Miyazawa, F.K., 2020. A robust optimization approach for the vehicle routing problem with selective backhauls. Transportation Research Part E: Logistics and Transportation Review 136, 101888.

Seyedhoseini, S., Rashid, R., Teimoury, E., 2015. Developing a cross-docking network design model under uncertain environment. Journal of Industrial Engineering International 11, 225–236.

Shaw, P., 1997. A new local search algorithm providing high quality solutions to vehicle routing problems. APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK 46.

Solano-Charris, E., Prins, C., Santos, A.C., 2015. Local search based metaheuristics for the robust vehicle routing problem with discrete scenarios. Applied Soft Computing 32, 518–531.
Sungur, I., Ordónez, F., Dessouky, M., 2008. A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty. IIE Transactions 40, 509–523.
Tarantilis, C.D., 2013. Adaptive multi-restart tabu search algorithm for the vehicle routing problem with cross-docking. Optimization Letters 7, 1583–1596.
Taş, D., Dellaert, N., Van Woensel, T., De Kok, T., 2013. Vehicle routing problem with stochastic travel times including soft time windows and service costs. Computers & Operations Research 40, 214–224.
Toth, P., Vigo, D., 2014. Vehicle Routing: Problems, Methods, and Applications. SIAM.
Touihri, A., Dridi, O., Krichen, S., 2016. A multi operator genetic algorithm for solving the capacitated vehicle routing problem with cross-docking problem, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE. pp. 1–8.
Van Belle, J., Valckenaers, P., Cattrysse, D., 2012. Cross-docking: State of the art. Omega 40, 827–846.
Van Breedam, A., 1995. Improvement heuristics for the vehicle routing problem based on simulated annealing. European Journal of Operational Research 86, 480–490.
Wang, A., Subramanyam, A., Gounaris, C.E., 2022. Robust vehicle routing under uncertainty via branch-price-and-cut. Optimization and Engineering 23, 1895–1948.
Wen, M., Larsen, J., Clausen, J., Cordeau, J.F., Laporte, G., 2009. Vehicle routing with cross-docking. Journal of the Operational Research Society 60, 1708–1718.
Yanıkoğlu, İ., Gorissen, B.L., den Hertog, D., 2019. A survey of adjustable robust optimization. European Journal of Operational Research 277, 799–813.
Yu, V.F., Jewpanya, P., Redi, A.P., 2016. Open vehicle routing problem with cross-docking. Computers & Industrial Engineering 94, 6–17.
Yu, V.F., Jewpanya, P., Redi, A.P., Tsao, Y.C., 2021. Adaptive neighborhood simulated annealing for the heterogeneous fleet vehicle routing problem with multiple cross-docks. Computers & Operations Research 129, 105205.
Zhang, Y., Guo, W., Negenborn, R.R., Atasoy, B., 2022. Synchromodal transport planning with flexible services: Mathematical model and heuristic algorithm. Transportation Research Part C: Emerging Technologies 140, 103711.

Appendix A. An example of cumulative loading computation

We describe the solution procedure using the following example. Consider the set of customers $C = \{1, 2, 3, 4\}$ and two budget sets $\mathscr{L} = \{1, 2\}$: $\mathscr{B}_1 = \{1, 2\}$ and $\mathscr{B}_2 = \{3, 4\}$. We aim to find value $\mathscr{Q}_{(\mathscr{R}_v)}$ where $\mathscr{R}_v = \{0, 1, 4, 2, 0\}$ and $C^- = \{1, 2, 4\}$. Following the schematic representation (see Figure 3), in this example the flow network is defined by a digraph including a set of vertices $\mathscr{V} = \{s, 1, 2, 4, \mathscr{B}_1, \mathscr{B}_2, t\}$ and a set of arcs $\mathscr{A} = \mathscr{A}_1 \cup \mathscr{A}_2 \cup \mathscr{A}_3 \cup \mathscr{A}_4$. In particular, $\mathscr{A}_1 = \{(s, 1), (s, 2), (s, 4)\}$ (blue arcs); $\mathscr{A}_2 = \{(1, t), (2, t), (4, t)\}$ (red arcs); $\mathscr{A}_3 = \{(1, \mathscr{B}_1), (2, \mathscr{B}_1), (4, \mathscr{B}_2)\}$ (green arcs); and $\mathscr{A}_4 = \{(\mathscr{B}_1, t), (\mathscr{B}_2, t)\}$ (black arcs) with corresponding capacities (see Figure A1(a)).

For the sake of description, we denote with $\mathscr{G}_r = (\mathscr{V}, \mathscr{A}_r)$ the *residual network*, where $\mathscr{A}_r$ is the set of arcs with their *residual capacities* $c_f(i, j)$ for all $(i, j) \in \mathscr{A}$ and with $\mathscr{P}$ an augmenting path with minimum residual capacity equal to $f(\mathscr{P})$. Flows on arcs are denoted by variables $f(i, j)$ such that $0 \le f(i, j) \le c(i, j)$. We then apply the Ford-Fulkerson algorithm to obtain the total flow (i.e., the maximum cumulative loading for the route $\mathscr{R}_v = \{0, 1, 4, 2, 0\}$) as shown in Table A1.

Table A1
Detailed results of the maximum flow problem of Figure 1(a)

Iter.	Path ($\mathscr{P}$)	Flow of path, $f(\mathscr{P})$	Flow and residual capacity of arcs	Total flow, $f(s\text{-}t)$
1	$\{s, 1, t\}$	$\underline{d}_1$	$f(s,1) = f(1,t) = \underline{d}_1$ $c_f(s,1) = \bar{d}_1 - \underline{d}_1 = \delta d_1,\ c_f(1,t) = 0$	$\underline{d}_1$
2	$\{s, 2, t\}$	$\underline{d}_2$	$f(s,2) = f(2,t) = \underline{d}_2$ $c_f(s,2) = \bar{d}_2 - \underline{d}_2 = \delta d_2,\ c_f(2,t) = 0$	$\underline{d}_1 + \underline{d}_2$
3	$\{s, 4, t\}$	$\underline{d}_4$	$f(s,4) = f(4,t) = \underline{d}_4$ $c_f(s,4) = \bar{d}_4 - \underline{d}_4 = \delta d_4,\ c_f(4,t) = 0$	$\underline{d}_1 + \underline{d}_2 + \underline{d}_4$
4	$\{s, 1, \mathscr{B}_1, t\}$	$x_1 = \min(\delta d_1, c_f(\mathscr{B}_1, t))$	$f(s,1) = \underline{d}_1 + x_1,\ f(1,\mathscr{B}_1) = f(\mathscr{B}_1,t) = x_1$ $c_f(s,1) = c_f(1,\mathscr{B}_1) = \delta d_1 - x_1,\ c_f(\mathscr{B}_1,t) = \beta_1 - x_1$	$\underline{d}_1 + \underline{d}_2 + \underline{d}_4 + x_1$
5	$\{s, 2, \mathscr{B}_1, t\}$	$x_2 = \min(\delta d_2, c_f(\mathscr{B}_1, t))$	$f(s,2) = \underline{d}_2 + x_2,\ f(2,\mathscr{B}_1) = x_2, f(\mathscr{B}_1,t) = x_1 + x_2$ $c_f(s,2) = c_f(2,\mathscr{B}_1) = \delta d_2 - x_2,\ c_f(\mathscr{B}_1,t) = \beta_1 - x_1 - x_2$	$\underline{d}_1 + \underline{d}_2 + \underline{d}_4 +$ $x_1 + x_2$
6	$\{s, 4, \mathscr{B}_2, t\}$	$x_4 = \min(\delta d_4, c_f(\mathscr{B}_2, t))$	$f(s,4) = \underline{d}_4 + x_4,\ f(4,\mathscr{B}_2) = f(\mathscr{B}_2,t) = x_4$ $c_f(s,4) = c_f(4,\mathscr{B}_2) = \delta d_4 - x_4,\ c_f(\mathscr{B}_2,t) = \beta_2 - x_4$	$\underline{d}_1 + \underline{d}_2 + \underline{d}_4 +$ $x_1 + x_2 + x_4$

Based on detailed results in Table A1, we note that the first term in Equation (48) is calculated by the first three iterations, while the terms of the minimization argument in Equation (48) are addressed in the remaining iterations.

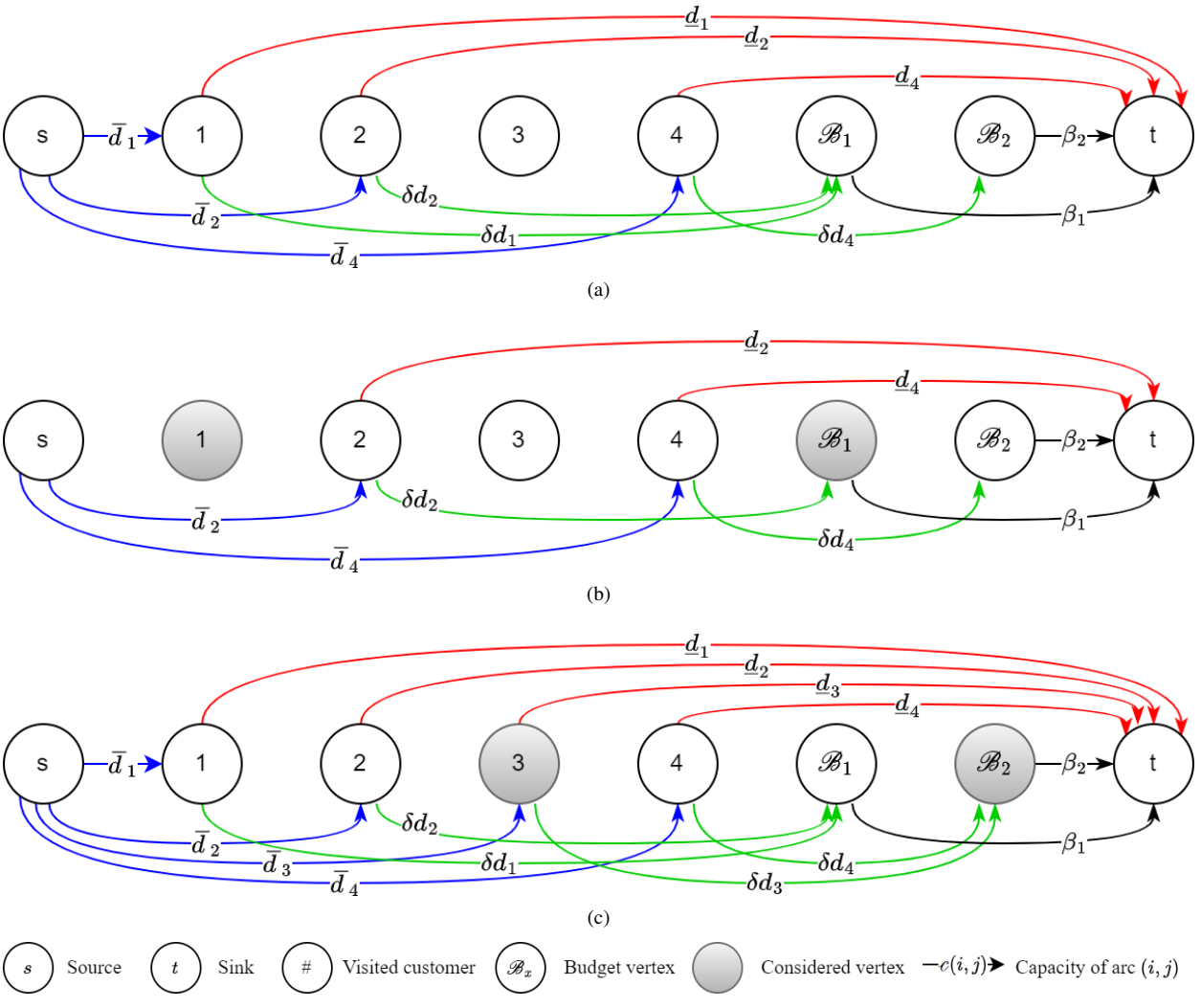


Figure A1: Illustrating the maximum flow network in three cases: (a) initial route $\mathscr{R}_v = \{0,1,4,2,0\}$; (b) Example 1: removing node 1, $\mathscr{R}_v = \{0,4,2,0\}$; and (c) Example 2: adding node 3, $\mathscr{R}_v = \{0,1,4,3,2,0\}$.

Updating a route based on a customer removal

Example 1 illustrates the procedure for updating the load when node 1 is removed from $\mathscr{R}_v$. First, arcs $(s,1)$, $(1,t)$, and $(1,\mathscr{B}_1)$ are removed as shown in Figure A1(b). Second, all paths relating node 1 and its budget $\mathscr{B}_1$ are considered, which are $\{s,\mathbf{1},t\}$, $\{s,\mathbf{1},\boldsymbol{\mathscr{B}_1},t\}$, and $\{s,2,\boldsymbol{\mathscr{B}_1},t\}$. The modifications are presented as follows.

- From path $\{s,\mathbf{1},t\}$, total flow is reduced by the flow of path $\underline{d}_1$.
- From path $\{s,\mathbf{1},\boldsymbol{\mathscr{B}_1},t\}$, the total of capacity arcs entering $\mathscr{B}_1$ is updated by $\rho_1^{after} = \rho_1^{before} - \delta d_1 = \delta d_2$. This implies Equation (49).
- From two paths $\{s,1,\boldsymbol{\mathscr{B}_1},\boldsymbol{t}\}$ and $\{s,2,\boldsymbol{\mathscr{B}_1},\boldsymbol{t}\}$, the total flow of arc $(\mathscr{B}_1,t)$ is updated by $\pi_1^{after} = \min(\rho_1^{after},\beta_1) = \min(\delta d_2,\beta_1)$. This implies Equation (50).
- The total flow is finally updated by using Equation (51). In particular, the total flow is reduced by $\underline{d}_1$ and also by the difference of the flow of $(\mathscr{B}_1,t)$ before and after the removal.

$$\begin{aligned} \mathbb{Q}_{(\mathscr{R}_v)}^{after} &= \mathbb{Q}_{(\mathscr{R}_v)}^{before} - \underline{d}_1 - \left[\pi_1^{before} - \pi_1^{after}\right] \\ &= (\underline{d}_1 + \underline{d}_2 + \underline{d}_4 + x_1 + x_2 + x_4) - \underline{d}_1 - \left[(x_1 + x_2) - \min(\delta d_2, \beta_1)\right] \end{aligned}$$

$$= \underline{d}_2 + \underline{d}_4 + x_4 + \min(\delta d_2, \beta_1)$$

Updating a route based on a customer insertion

Example 2 similarly illustrates the procedure for updating the load when node 3 is inserted into $\mathscr{R}_v$. Arcs $(s,3)$, $(3,t)$, and $(3,\mathscr{B}_2)$ are inserted as shown in Figure A1(c). Next, all paths relating node 3 and its budget $\mathscr{B}_2$ are considered, which are $\{s,\mathbf{3},t\}$, $\{s,\mathbf{3},\boldsymbol{\mathscr{B}_2},t\}$, and $\{s,4,\boldsymbol{\mathscr{B}_2},t\}$. The modifications are presented as follows.

- From path $\{s,\mathbf{3},t\}$, total flow is increased by the flow of path $\underline{d}_3$.
- From path $\{s,\mathbf{3},\boldsymbol{\mathscr{B}_2},t\}$, the total of capacity arcs entering $\mathscr{B}_2$ is updated by $\rho_2^{after} = \rho_2^{before} + \delta d_3 = \delta d_3 + \delta d_4$. This implies Equation (52).
- From two paths $\{s,3,\boldsymbol{\mathscr{B}_2},\boldsymbol{t}\}$ and $\{s,4,\boldsymbol{\mathscr{B}_2},\boldsymbol{t}\}$, the total flow of arc $(\mathscr{B}_2,t)$ is updated by $\pi_2^{after} = \min(\rho_2^{after}, \beta_2) = \min(\delta d_3 + \delta d_4, \beta_2)$. This implies Equation (53).
- The total flow is then updated by using Equation (54). In particular, the total flow is increased by $\underline{d}_3$ and also by the difference of the flow of $(\mathscr{B}_2,t)$ after and before the insertion.

$$\begin{aligned}
\mathbb{Q}_{(\mathscr{R}_v)}^{after} &= \mathbb{Q}_{(\mathscr{R}_v)}^{before} + \underline{d}_3 + \left[\pi_2^{after} - \pi_2^{before}\right] \\
&= (\underline{d}_1 + \underline{d}_2 + \underline{d}_4 + x_1 + x_2 + x_4) + \underline{d}_3 + \left[\min(\delta d_3 + \delta d_4, \beta_2) - x_4\right] \\
&= \underline{d}_1 + \underline{d}_2 + \underline{d}_3 + \underline{d}_4 + x_1 + x_2 + \min(\delta d_3 + \delta d_4, \beta_2)
\end{aligned}$$

Appendix B. An example of solving the problem under demand uncertainty

This section gives an example analyzing the impact of demand uncertainty. We consider instance 10-4 whose optimal solutions are obtained using the MIP model. In addition, three cases with different uncertainty levels (see Table C5) are considered to show how the uncertainty impacts the routing plan and the total cost. The three cases are defined as follows.

- Case 1: the uncertainty level 0 $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma}) = (0,0)$, which represents the deterministic problem.
- Case 2: the uncertainty level 7 $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma}) = (0.4, 0.25)$.
- Case 3: the uncertainty level 14 $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma}) = (0.6, 0.75)$, which represents a higher level than Case 2.

In addition, for each uncertainty level we use a feasible demand realization (see Table B2) to discuss the robustness of the solutions, as shown in the following.

Table B2
A sample of demand realizations under three cases for instance 10-4

	Deterministic demand (Case 0)				Demand realization at level 7 (Case 7)				Demand realization at level 14 (Case 14)			
	Product 1	Product 2	Product 3	Product 4	Product 1	Product 2	Product 3	Product 4	Product 1	Product 2	Product 3	Product 4
Customer 1	0	0	4	13	0	0	4	14	0	0	5	18
Customer 2	11	5	1	0	12	5	1	0	15	7	1	0
Customer 3	17	0	0	0	18.5	0	0	0	24	0	0	0
Customer 4	10	0	0	0	11	0	0	0	14	0	0	0
Customer 5	0	0	0	18	0	0	0	19.5	0	0	0	25
Customer 6	0	0	0	15	0	0	0	16	0	0	0	21
Total supply	38	5	5	46	41.5	5	5	49.5	53	7	6	64

To solve a problem under demand uncertainty using RO, we highlight some points as follows.

- First, we define the uncertainty set by using Equation (17) based on two scalar parameters $(\boldsymbol{\varepsilon},\boldsymbol{\Gamma})$ and customer subsets $\mathscr{B}_l \; \forall \in 1,\ldots,L$ (see Section 4.1). In this example (i.e., instance 10-4), customer subsets are $\mathscr{B}_1 = \{1,3\}$, $\mathscr{B}_2 = \{4,5\}$, and $\mathscr{B}_3 = \{2,6\}$, obtained by using the "Clustered" method (P_2) (see Section 6.1).

- Second, for each case with the uncertainty set defined above, we solve to optimality by using the MIP model. Results show that, as expected, the higher the uncertainty level is, the higher is the cost that arises (i.e., the total cost of three cases 1, 2, and 3 are 7229, 7255, and 8710, respectively). In Cases 2 and 3, additional costs are caused by the modification of supplier and customer routes to handle demand uncertainty.
- More specifically, Figure B2 and Table B3 analyze three solutions under three uncertainty levels. The figure shows how blue routes are adjusted to make solutions more robust and how red routes are infeasible routes under some demand realizations and worst-case realizations regarding uncertainty levels.
- Table B3 shows that solution (a) is infeasible for both levels 7 and 14, whereas solution (b) is infeasible only for level 14 (under the generated demand realizations and the worst-case realizations). However, solution (c) is feasible at all three levels. We therefore state that solution (b) is more robust than solution (a), whereas solution (c) is the most robust solution. Of course, robustness is achieved at an additional cost.
- As shown in Figure B2, solutions (a), (b), and (c) are the best solutions for handling the uncertainty levels 0, 7, and 14, respectively (see green checkmarks).

Table B3
Detailed results of the optimal routing plans under three different uncertainty levels for instance 10-4

Solution	Route		Sequence	Nominal Loading	A sample loading		Worst-case loading		Unmet quantity	
					At level 7	At level 14	At level 7	At level 14	At level 7	At level 14
Solution (a)	Customer	Route 1	0 C3 C1 C6 C2 0	66	**70.5**	**91**	**72.6**	**95.7**	2.6	25.7
		Route 2	0 C5 C4 0	28	30.5	39	30.8	40.6	0	0
	Supplier	Route 1	0 S2 S4 S3 0	56	59.5	**77**	61.6	**81.2**	0	11.2
		Route 2	0 S1 0	38	41.5	53	41.8	55.1	0	0
Solution (b)	Customer	Route 1	0 C5 C4 C3 C1 0	62	67	**86**	68.2	**89.9**	0	19.9
		Route 2	0 C6 C2 0	32	34	44	35.2	46.4	0	0
	Supplier	Route 1	0 S2 S4 S3 0	56	59.5	**77**	61.6	**81.2**	0	11.2
		Route 2	0 S1 0	38	41.5	53	41.8	55.1	0	0
Solution (c)	Customer	Route 1	0 C3 C1 0	34	36.5	47	37.4	49.3	0	0
		Route 2	0 C5 C6 C4 0	43	46.5	60	47.3	62.35	0	0
		Route 3	0 C2 0	17	18	23	18.7	25.65	0	0
	Supplier	Route 1	0 S2 S3 S1 0	48	51.5	66	52.8	69.6	0	0
		Route 2	0 S4 0	46	49.5	64	50.6	66.7	0	0

***Bold values** exceed the vehicle capacity $Q = 70$.

Appendix C. Detailed computational results

Table C4
Tuned parameters in the adaptive weight adjustment

Notation	Description
γ	Reaction factor controls how quickly the weight adjustment is
ζ_1	Adding score to destroy/repair operators if a new best solution is found
ζ_2	Adding score to destroy/repair operators if an incumbent solution is improved
ζ_2	Adding score to destroy/repair operators if a new solution is worse than the current solution
ω_0	Initial weight of destroy and repair operators

Table C5
All scenarios of the uncertainty levels $(\boldsymbol{\varepsilon}, \boldsymbol{\Gamma})$

Level	ε	Γ	Level	ε	Γ	Level	ε	Γ
1	0.2	0	6	0.4	0	11	0.6	0
2	0.2	0.25	7	0.4	0.25	12	0.6	0.25
3	0.2	0.5	8	0.4	0.5	13	0.6	0.5
4	0.2	0.75	9	0.4	0.75	14	0.6	0.75
5	0.2	1	10	0.4	1	15	0.6	1

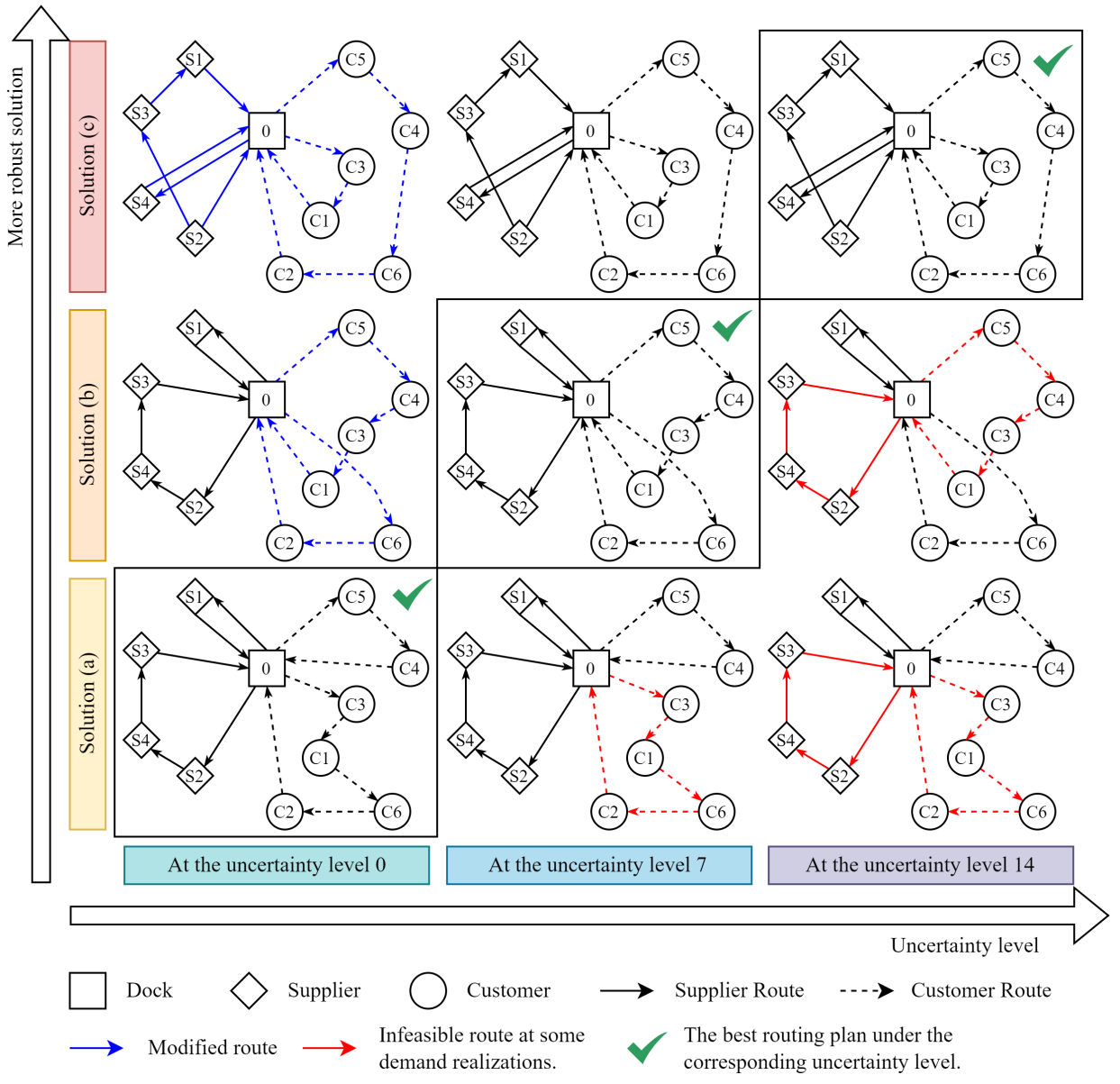


Figure B2: Graphical representation of the three solutions (a), (b), and (c) under three uncertainty levels.

Table C6
Tuning parameters

(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
T_0	200, 400, 600, 800, 1000	-	800	ζ_1	100, 66, 33	33	33
α	0.7, 0.8, 0.9, 0.95	0.9	0.95	ζ_2	33, 33, 17	17	17
θ	30, 40, 50, 60	50	40	ζ_3	33, 17, 5	5	5
η_{ALNS}	50, 100, 150, 200	150	150	ω_0	5, 10, 20	5	5
η_{Cool}	50, 100, 150, 200	150	150	ϕ_1	0.33, 0.001, 0.0005	0.0005	0.0005
η_{min}	3, 5, 7	5	5	ϕ_2	0.33, 0.0025, 0.0025	0.0025	0.0025
η_{max}	7, 10, 12	12	12	ϕ_3	0.33, 0.25, 0.25	0.25	0.25
γ	0.1, 0.2, 0.3	0.2	0.2	p_{worst}	3, 4, 5, 6	3	6
				y_{noise}	5, 20, 30, 100	20	30

(1) Parameters, (2) Range Value, (3) Initial value, and (4) Tuned value.

Table C7
Computational results of the deterministic VRPCD compared to state-of-art algorithms in Set 1 instances

Instance	Optimal	TS		imp-TS		SA		Matheuristic			ALNS			GAP (%) of Avg. TC					GAP (%) of Best	
		Avg.	CPU	Avg.	CPU	Avg.	CPU	Avg.	Best	CPU	Avg.	Best	CPU	TS	imp-TS	SA	Math.	ALNS	Math.	ALNS
1	6823	7571.4	1.52	6847.6	0.22	6953	1.91	6823	6823	0.26	6823	6823	0.07	10.97	0.36	1.91	0.00	0	0	0
2	6741	7103.7	1.74	6816.8	0.23	6741	1.78	6741	6741	0.19	6741	6741	0.06	5.38	1.12	0.00	0.00	0	0	0
3	9269	9993.5	2.37	9615.6	0.19	9269	2.44	9269	9269	0.18	9269	9269	0.07	7.82	3.74	0.00	0.00	0	0	0
4	7229	8338	1.6	7289.7	0.27	7255	1.75	7229	7229	0.18	7229	7229	0.06	15.34	0.84	0.36	0.00	0	0	0
5	6475	8709.9	2.28	6599	0.21	6524	1.75	6475	6475	0.18	6475	6475	0.06	34.52	1.92	0.76	0.00	0	0	0
6	7434	9143.5	1.82	9324.6	0.03	7613	1.81	7434	7434	0.13	7434	7434	0.06	23.00	25.43	2.41	0.00	0	0	0
7	11713	12721.2	2.8	12083	0.01	11990	2.77	11713	11713	0.14	11713	11713	0.07	8.61	3.16	2.36	0.00	0	0	0
8	8158	9275.7	1.85	8719.6	0.04	8158	2.21	8158	8158	0.16	8158	8158	0.06	13.70	6.88	0.00	0.00	0	0	0
9	6989	8096.5	2.04	7362.2	0.25	7120	1.75	6989	6989	0.15	6989	6989	0.06	15.85	5.34	1.87	0.00	0	0	0
10	5960	7044.8	1.82	6204.5	0.36	6056	2.05	5960	5960	0.16	5960	5960	0.06	18.20	4.10	1.61	0.00	0	0	0
11	6916	8051.8	1.8	7635.3	0	7434	1.93	6916	6916	0.14	6916	6916	0.06	16.42	10.40	7.49	0.00	0	0	0
12	7656	8661	1.72	7867.2	0.24	7800	1.76	7656	7656	0.16	7656	7656	0.06	13.13	2.76	1.88	0.00	0	0	0
13	6783	7370.2	1.54	7097.9	0.24	6934	1.88	6783	6783	0.16	6783	6783	0.06	8.66	4.64	2.23	0.00	0	0	0
14	4417	7132.3	1.53	5208	0	4704	1.69	4417	4417	0.15	4417	4417	0.05	61.47	17.91	6.50	0.00	0	0	0
15	7072	7563.4	1.61	7103.2	0.41	7088	1.75	7072	7072	0.15	7072	7072	0.06	6.95	0.44	0.23	0.00	0	0	0
16	8440	9983.6	2.05	8768.7	0.03	8616	2.05	8440	8440	0.13	8440	8440	0.06	18.29	3.89	2.09	0.00	0	0	0
17	9003	9538.1	2.26	9003	0.06	9003	2.6	9003	9003	0.15	9003	9003	0.07	5.94	0.00	0.00	0.00	0	0	0
18	6760	8057.4	1.74	6887.5	0.19	6911	1.88	6760	6760	0.15	6760	6760	0.06	19.19	1.89	2.23	0.00	0	0	0
19	7051	9042.6	2.21	7123	0.03	7051	1.92	7051	7051	0.13	7051	7051	0.06	28.25	1.02	0.00	0.00	0	0	0
20	9786	10478	2.55	10471	0	10004	2.32	9786	9786	0.14	9786	9786	0.07	7.07	7.00	2.23	0.00	0	0	0
21	4644	8380.5	2.06	5431.4	0	4753	1.51	4644.2	4644	0.16	4644	4644	0.06	80.46	16.96	2.35	0.00	0	0	0
22	6442	9016.9	2.42	6908	0.01	6442	1.85	6442	6442	0.13	6442	6442	0.06	39.97	7.23	0.00	0.00	0	0	0
23	9156	9489.2	2.31	9224.1	0.11	9156	2.28	9156	9156	0.16	9156	9156	0.06	3.64	0.74	0.00	0.00	0	0	0
24	11976	12513.6	2.64	11976	0	11976	2.78	11976	11976	0.12	11976	11976	0.07	4.49	0.00	0.00	0.00	0	0	0
25	6346	7114.3	1.68	6638	0.1	6346	1.94	6346	6346	0.16	6346	6346	0.06	12.11	4.60	0.00	0.00	0	0	0
26	6817	8421.3	2.04	7216.9	0.03	6880	1.86	6817	6817	0.16	6817	6817	0.06	23.53	5.87	0.92	0.00	0	0	0
27	9541	10666.8	2.47	9709.8	0.06	9541	2.39	9541	9541	0.15	9541	9541	0.06	11.80	1.77	0.00	0.00	0	0	0
28	6782	10123.3	2.69	7408	0.01	7107	1.84	6782	6782	0.13	6782	6782	0.06	49.27	9.23	4.79	0.00	0	0	0
29	6591	7503.2	1.73	6748.5	0.18	6762	1.86	6591	6591	0.17	6591	6591	0.06	13.84	2.39	2.59	0.00	0	0	0
30	6919	7642.6	1.82	7304.4	0.09	6942	3.51	6919	6919	0.16	6919	6919	0.06	10.46	5.57	0.33	0.00	0	0	0
Average			**2.02**		**0.12**		**2.06**			**0.16**			**0.06**	**19.61**	**5.24**	**1.57**	**0**	**0**		**0**

Table C8
Computational results of the deterministic VRPCD compared to state-of-art algorithms in Set 2 instances

Instance	BKS	TS		imp-TS		SA		Matheuristic			ALNS			GAP (%) of Avg. TC					GAP (%) of Best	
		Avg.	CPU	Avg.	CPU	Avg.	CPU	Avg.	Best	CPU	Avg.	Best	CPU	TS	imp-TS	SA	Math.	ALNS	Math.	ALNS
1	6620	12366.7	3	7692.9	0.37	7550.2	2.8	6620	6401	1.87	6476.8	6426	1.07	86.81	16.21	14.05	0	-2.16	0	0.39
2	6658.2	14173	3.55	7787.2	0.16	7832.7	2.9	6658.2	6593	1.77	6559.6	6519	1.28	112.87	16.96	17.64	0	-1.48	0	-1.12
3	6626.3	13836.8	4.32	7893.6	0.43	7747.4	3	6626.3	6510	1.82	6475.8	6369	1.20	108.82	19.13	16.92	0	-2.27	0	-2.17
4	6377.7	10995.4	2.09	7792.2	0.23	7677.4	3	6377.7	6259	1.36	6299.8	6259	1.07	72.40	22.18	20.38	0	-1.22	0	0.00
5	6289.6	11757.8	2.26	7224.8	0.39	7579.7	3.5	6289.6	6105	2.01	6249.2	6105	1.14	86.94	14.87	20.51	0	-0.64	0	0.00
6	5656.6	11027.7	2.1	7245.9	0.22	7053	3.1	5656.6	5617	1.21	5641.2	5617	0.95	94.95	28.10	24.69	0	-0.27	0	0.00
7	6865.1	11899.2	2.61	8206.9	0.11	7720.1	3	6865.1	6679	2.41	6680.4	6615	1.20	73.33	19.55	12.45	0	-2.69	0	-0.96
8	6606.3	12825.5	3	7880.9	0.16	7709.8	3.3	6606.3	6351	2.04	6584.6	6421	1.20	94.14	19.29	16.70	0	-0.33	0	1.10
9	6820.1	12718.6	3.1	8157.3	0.16	7882.5	2.8	6820.1	6569	1.54	6502.6	6469	1.04	86.49	19.61	15.58	0	-4.66	0	-1.52
10	6729	11794.7	2.38	7924.7	0.2	7734.9	2.6	6729	6492	1.77	6599.6	6541	1.16	75.28	17.77	14.95	0	-1.92	0	0.75
11	6421.3	12094.9	3.03	7452.6	0.29	7721.7	3	6421.3	6322	1.26	6339	6290	1.03	88.36	16.06	20.25	0	-1.28	0	-0.51
12	6698.1	12132.5	2.64	8320	0.16	7899.8	2.9	6698.1	6631	1.49	6691.4	6611	0.95	81.13	24.21	17.94	0	-0.10	0	-0.30
13	6766.3	13223.4	2.92	8222.7	0.15	7863.7	2.9	6766.3	6628	1.63	6712.4	6575	1.06	95.43	21.52	16.22	0	-0.80	0	-0.80
14	6775.4	12413.9	2.76	8211.7	0.18	8141.1	3.6	6775.4	6695	1.44	6631.8	6561	1.13	83.22	21.20	20.16	0	-2.12	0	-2.00
15	6711.4	12521.4	3.06	8144.6	0.38	7941.6	3.1	6711.4	6535	1.67	6693.8	6655	1.09	86.57	21.35	18.33	0	-0.26	0	1.84
16	6763.9	12044.4	3.15	7451.7	0.28	7901.9	3.1	6763.9	6719	1.58	6653.8	6603	1.42	78.07	10.17	16.82	0	-1.63	0	-1.73
17	6709.7	12699.4	2.14	8086.2	0.34	8055	3	6709.7	6612	1.43	6617.4	6559	0.96	89.27	20.52	20.05	0	-1.38	0	-0.80
18	6655.3	11001.4	1.7	7576	0.28	7798.3	2.9	6655.3	6541	1.53	6581.6	6532	1.13	65.30	13.83	17.17	0	-1.11	0	-0.14
19	6788.5	12724.4	2.77	7871.2	0.36	7964.3	3.1	6788.5	6562	1.8	6638.8	6562	1.13	87.44	15.95	17.32	0	-2.21	0	0.00
20	6775.9	12357.7	2.72	7883.7	0.24	7522.4	2.5	6775.9	6382	2.06	6409.6	6369	1.31	82.38	16.35	11.02	0	-5.41	0	-0.20
21	6655.6	13177	3.39	7914.1	0.26	7886.2	3.4	6655.6	6489	1.49	6592.2	6549	1.03	97.98	18.91	18.49	0	-0.95	0	0.92
22	6694.7	11545	2.43	8005.3	0.35	7841.1	2.9	6694.7	6566	1.55	6584.2	6511	0.98	72.45	19.58	17.12	0	-1.65	0	-0.84
23	6574.4	12308.1	2.93	7883.5	0.38	7791.5	3.2	6574.4	6479	1.58	6501	6446	1.24	87.21	19.91	18.51	0	-1.12	0	-0.51
24	6611.9	12722.7	2.87	7731.2	0.48	7957.8	3	6611.9	6537	1.56	6561.2	6539	1.18	92.42	16.93	20.36	0	-0.77	0	0.03
25	6766.7	12844.9	2.67	7884.8	0.2	7839.4	3	6766.7	6490	1.55	6600.4	6490	1.08	89.83	16.52	15.85	0	-2.46	0	0.00
26	6956.1	13297.5	3.31	8001.6	0.16	7846.2	3.3	6956.1	6703	2.36	6759.4	6696	1.07	91.16	15.03	12.80	0	-2.83	0	-0.10
27	7164.1	13415.2	3.25	8899.4	0.17	8128.6	3.3	7164.1	6733	1.64	6917.2	6843	1.21	87.26	24.22	13.46	0	-3.45	0	1.63
28	7266.2	12613	2.6	10131	0	8367.5	2.8	7266.2	7191	2.09	7232.6	7132	1.09	73.58	39.43	15.16	0	-0.46	0	-0.82
29	6900.8	12840.8	3.28	8276.9	0.38	8003.1	2.9	6900.8	6692	1.9	6658.2	6550	1.26	86.08	19.94	15.97	0	-3.52	0	-2.12
30	6630.5	13796.2	3.78	8251.6	0.28	7760.9	2.7	6630.5	6520	1.66	6580	6510	1.15	108.07	24.45	17.05	0	-0.76	0	-0.15
Avg.			**2.86**		**0.26**		**3.02**			**1.7**			**1.13**	**87.17**	**19.66**	**17.13**	**0**	**-1.73**		**-0.34**

Table C9
Computational results of the deterministic VRPCD compared to state-of-art algorithms in Set 3 instances

Instance	BKS	TS		imp-TS		SA		Matheuristic			ALNS			GAP (%) of Avg. TC					GAP (%) of Best	
		Avg.	CPU	Avg.	CPU	Avg.	CPU	Avg.	Best	CPU	Avg.	Best	CPU	TS	imp-TS	SA	Math.	ALNS	Math.	ALNS
1	15905.4	24284.6	5.62	20704.6	0.49	19804.5	7.49	15905.4	15731	7.06	15721.6	15099	2.98	52.68	30.17	24.51	0	-1.16	0	-4.02
2	14110.1	23435.6	5.73	20816.8	0.64	18248.1	6.62	14110.1	13687	6.31	13595.4	13304	3.07	66.09	47.53	29.33	0	-3.65	0	-2.80
3	13872	23449.4	5.8	19612.2	0.3	18133.9	7.1	13872	13554	5.89	13534.6	13242	3.47	69.04	41.38	30.72	0	-2.43	0	-2.30
4	15087.1	23471.1	5.87	19549	0.44	19083.6	6.89	15087.1	14624	5.74	14535.2	14124	3.95	55.57	29.57	26.49	0	-3.66	0	-3.42
5	14664.3	23406.2	7.28	20448	0.61	18877.3	7.09	14664.3	14091	6.74	14386.8	13801	3.79	59.61	39.44	28.73	0	-1.89	0	-2.06
6	15206.3	24026.6	5.38	21212	0.56	19783	7.47	15206.3	14240	8.26	14585.8	14375	3.13	58.00	39.49	30.10	0	-4.08	0	0.95
7	15946.7	24190	7.65	20640.2	0.55	19690	6.11	15946.7	15689	7.06	15803.4	15450	3.56	51.69	29.43	23.47	0	-0.90	0	-1.52
8	14643.2	23158.9	9.88	20664.1	0.42	18939.2	6.91	14643.2	14108	5.14	14103.4	13715	3.39	58.15	41.12	29.34	0	-3.69	0	-2.79
9	14163.4	23594.7	5.54	18920	0.38	18510.6	7.11	14163.4	14009	6.01	13897	13618	3.34	66.59	33.58	30.69	0	-1.88	0	-2.79
10	15754.5	23530.5	5.77	20384.2	0.52	19607.3	7.1	15754.5	15248	7	15469.8	15143	3.60	49.36	29.39	24.46	0	-1.81	0	-0.69
11	14703.5	23371.7	5.37	19941.6	0.37	18675.8	7.15	14703.5	14368	6.98	14537.8	14278	2.82	58.95	35.62	27.02	0	-1.13	0	-0.63
12	13126.3	21082.8	4.46	17258.4	0.17	17550.3	6.75	13126.3	12710	5.66	12830.6	12722	4.04	60.61	31.48	33.70	0	-2.25	0	0.09
13	13366.3	21610.7	4.62	17829.9	0.16	18039.2	6.32	13366.3	12559	6.03	12686.2	12559	3.29	61.68	33.39	34.96	0	-5.09	0	0.00
14	13685.6	23397.9	5.44	19845.2	0.53	18252.5	7.38	13685.6	13190	5.87	13331.8	13177	3.13	70.97	45.01	33.37	0	-2.59	0	-0.10
15	15445.9	24041.9	6.31	21863	0.55	19803.8	7.97	15445.9	15015	6.99	15264.2	14872	3.01	55.65	41.55	28.21	0	-1.18	0	-0.95
16	14304.4	22893.4	5.18	20144.2	0.3	18808.3	6.95	14304.4	13614	6.79	14211.2	13888	3.04	60.04	40.83	31.49	0	-0.65	0	2.01
17	14628	22950.4	6.93	20093.3	0.44	18713.8	6.72	14628	13853	5.48	14159.6	13959	3.50	56.89	37.36	27.93	0	-3.20	0	0.77
18	14291.9	24358.2	6.25	20244.8	0.53	18579.7	7.07	14291.9	13683	7.85	14184.4	13740	3.30	70.43	41.65	30.00	0	-0.75	0	0.42
19	14956	25068.7	5.68	19955	0.28	18453.2	7.93	14956	14554	7.16	14372.6	13794	3.67	67.62	33.42	23.38	0	-3.90	0	-5.22
20	13932.5	23232.1	4.79	19267.7	0.36	18167.3	7.56	13932.5	13518	5.99	13618.6	13069	3.61	66.75	38.29	30.40	0	-2.25	0	-3.32
21	14264.4	22564.8	5.43	19533.4	0.61	19226	6.97	14264.4	13838	6.83	14019.2	13948	4.18	58.19	36.94	34.78	0	-1.72	0	0.79
22	14170.5	24360.7	6.04	19032.1	0.37	18551.6	7.83	14170.5	13740	6.35	13763.4	13629	2.78	71.91	34.31	30.92	0	-2.87	0	-0.81
23	14463.9	24377.9	5.88	20562.5	0.51	18514.8	7.35	14463.9	14175	5.68	14142.2	13678	3.47	68.54	42.16	28.01	0	-2.22	0	-3.51
24	13589.9	22008.7	5.36	19288.2	0.05	18558.5	6.81	13589.9	13185	5.76	13517.6	13287	3.15	61.95	41.93	36.56	0	-0.53	0	0.77
25	14512.4	24256.6	5.76	19695.9	0.33	18574.2	7.55	14512.4	14212	6.78	14253	13624	4.45	67.14	35.72	27.99	0	-1.79	0	-4.14
26	14554.5	23424.9	5.05	20610.5	0.14	18995.7	7.8	14554.5	13871	6.31	13906.8	13468	3.71	60.95	41.61	30.51	0	-4.45	0	-2.91
27	14571.2	22961.4	5.17	18942.8	0.31	18128.7	7.12	14571.2	14300	7.41	14307.6	13783	3.66	57.58	30.00	24.41	0	-1.81	0	-3.62
28	14332.3	23822.3	5.56	20097.3	0.39	18952.4	7.41	14332.3	13875	6.11	13964.2	13771	2.94	66.21	40.22	32.24	0	-2.57	0	-0.75
29	14713.1	23678.3	64.84	22248.1	0.2	19056.1	6.96	14713.1	13948	5.19	14535.8	14004	3.16	60.93	51.21	29.52	0	-1.21	0	0.40
30	14451.7	23149.8	5.91	19321.9	0.65	18268.6	8.29	14451.7	14022	5.83	14024.8	13664	3.33	60.19	33.70	26.41	0	-2.95	0	-2.55
Avg.			**7.82**		**0.41**		**7.19**			**6.41**			**3.42**	**61.67**	**37.58**	**29.32**	**0**	**-2.34**		**-1.49**

Table C10
The computational environment of the state-of-art algorithms

Algorithm	Machine	Programming	Single thread rating
Lee et al. (2006)	No information	No information	549 (Assumed)
Liao et al. (2010)	Intelő Pentium 4 @ 3.2 GHz	No information	549
Yu et al. (2016)	Intelő Core i7-6700 CPU @ 3.40GHz	C++	2302
Gunawan et al. (2021)	Intelő Core i7-8700 CPU @ 3.20GHz	C++	2674
Our research	Intelő Core i7-10700 CPU @ 2.90GHz	C++	2930

Table C11
Computational results of VRPCD-DU compared to the commercial solver (GUROBI)

Instance	Set 1						Set 2						Set 3					
	GUROBI		ALNS				GUROBI		ALNS				GUROBI		ALNS			
	Obj.	CPU	Best	Avg.	CPU	GAP	Obj.	CPU	Best	Avg.	CPU	GAP	Obj.	CPU	Best	Avg.	CPU	GAP
1	8702	98.78	8702	8702	0.081	0	9043	14400	7396	7457.6	1.71	-18.21	-	28800	15344	15480.0	4.17	-
2	6913	65.83	6913	6913	0.0856	0	9171	14400	7397	7439.8	2.01	-19.34	-	28800	13930	14192.4	3.57	-
3	11245	9.19	11245	11245	0.0844	0	9063	14400	7360	7379.6	1.79	-18.79	-	28800	13756	13993.0	4.32	-
4	7412	75.78	7412	7412	0.0744	0	9806	14400	7249	7294.4	1.28	-26.08	-	28800	14449	14876.4	3.47	-
5	7011	48.53	7011	7011	0.0768	0	8166	14400	7145	7201.8	1.53	-12.50	-	28800	14398	14658.0	3.85	-
6	10069	40.77	10069	10069	0.0774	0	8256	14400	6702	6718	1.40	-18.82	-	28800	14526	14944.0	4.11	-
7	16586	7.03	16586	16586	0.0926	0	9279	14400	7235	7295	1.50	-22.03	-	28800	15912	16092.4	3.25	-
8	10832	24.89	10832	10832	0.0776	0	8792	14400	7066	7122.4	1.63	-19.63	-	28800	14656	14760.0	4.00	-
9	7337	44.83	7337	7337	0.0722	0	9210	14400	6469	6498.8	1.30	-29.76	-	28800	13760	13997.6	3.67	-
10	6157	113.99	6157	6157	0.0722	0	9054	14400	7241	7256	1.62	-20.02	-	28800	15338	15621.0	4.14	-
11	9291	38.23	9291	9291	0.0798	0	8581	14400	7003	7020	1.53	-18.39	-	28800	14188	14597.8	3.25	-
12	8003	63.26	8003	8003	0.0746	0	8990	14400	7459	7542.8	1.29	-17.03	-	28800	12737	12909.6	3.41	-
13	7075	68.06	7075	7075	0.0718	0	10475	14400	7751	7796.2	1.41	-26.00	-	28800	12908	13153.0	3.59	-
14	4417	18.16	4417	4417	0.0654	0	8530	14400	6616	6645.8	1.50	-22.44	-	28800	13202	13551.6	3.36	-
15	7202	93.64	7202	7202	0.0734	0	8299	14400	7329	7391.8	1.43	-11.69	-	28800	14887	15261.4	3.00	-
16	10046	25.16	10046	10046	0.0794	0	9143	14400	7638	7649.8	1.89	-16.46	-	28800	13838	14165.6	3.64	-
17	12126	5.88	12126	12126	0.0844	0	9814	14400	6575	6599.2	1.30	-33.00	-	28800	13927	14460.2	3.56	-
18	8959	46.59	8959	8959	0.0774	0	8865	14400	7488	7529	1.51	-15.53	-	28800	13937	14267.0	4.27	-
19	10267	30.5	10267	10267	0.08	0	9107	14400	7256	7308	1.64	-20.33	-	28800	14457	14669.0	3.99	-
20	12869	2.11	12869	12869	0.0852	0	7861	14400	6327	6413.2	1.58	-19.51	-	28800	13402	13627.4	3.40	-
21	4644	24.89	4644	4644	0.0694	0	9260	14400	7254	7292.4	1.57	-21.66	-	28800	14093	14401.4	3.37	-
22	9499	33.06	9499	9499	0.0794	0	8607	14400	6510	6568.2	1.59	-24.36	-	28800	14052	14160.6	3.78	-
23	12319	4.2	12319	12319	0.08	0	9003	14400	7444	7472.2	1.36	-17.32	-	28800	14563	15103.8	3.94	-
24	15482	0.94	15482	15482	0.0848	0	8908	14400	6536	6544.8	1.47	-26.63	-	28800	13595	13817.8	3.46	-
25	9025	11.36	9025	9025	0.0804	0	9447	14400	7360	7402	1.53	-22.09	-	28800	14300	14586.2	3.29	-
26	9707	10.91	9707	9707	0.081	0	9877	14400	7532	7584.6	1.53	-23.74	-	28800	13940	14188.8	3.63	-
27	10659	2.83	10659	10659	0.0822	0	10308	14400	7594	7642.2	1.65	-26.33	-	28800	14107	14315.8	3.48	-
28	9894	10.59	9894	9894	0.0818	0	9229	14400	7811	7849.2	1.51	-15.36	-	28800	14155	14322.2	3.44	-
29	7060	25.7	7060	7060	0.0764	0	9418	14400	7589	7690	1.36	-19.42	-	28800	14077	14400.6	3.71	-
30	9912	32.66	9912	9912	0.0798	0	8297	14400	6494	6568	1.41	-21.73	-	28800	13810	14181.6	3.12	-
Average		**35.95**			**0.08**	**0**		**14400**			**1.53**	**-20.81**		**28800**			**3.64**	-

Table C12
Computational results of the ALNS algorithm for solving VRPCD-DU in Set 1 instances

Set 1	Combination															
Instance	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	6823	6823	6823	7079	7115	7163	6823	7111	7163	8509	9570	6823	7163	8702	9681	9711
2	6741	6741	6742	6742	6742	6806	6741	6742	6818	6841	8032	6741	6742	6913	9162[a]	9162[a]
3	9269	9552	9599	9599	9726	9726	9552	9599	9828	9862	12476	9699	9828	11245	12580	14352[a]
4	7229	7229	7229	7255	7309	7309	7229	7255	7309	7412	7739	7229	7309	7412	8710	10148[a]
5	6475	6475	6475	6494	6494	6494	6475	6494	6494	7011	9180	6475	6494	7011	9180	9180
6	7434	7434	8711	10069	10069	10069	7434	10069	10069	10069	10069	7434	10069	10069	10069	11998[a]
7	11713	11849	13198	13198	13198	13198	13198	13198	13198	15058	15058	13198	13198	16586	16586[a]	16586[a]
8	8158	9382	9382	9382	9475	10832	9382	9382	10832	10832	10832	9382	9475	10832	12446	12515[a]
9	6989	6989	7156	7156	7156	7325	6989	7156	7325	7337	7340	6989	7156	7337	8651	9860
10	5960	5960	5960	6157	6157	6157	5960	6157	6157	6157	6577	5960	6157	6157	6683	6683[a]
11	6916	6916	7075	9171	9171	9291	6916	9171	9291	9291	9291	6916	9291	9291	9291	10801
12	7656	7656	7656	7784	7784	7784	7656	7784	7784	8003	8003	7656	7784	8003	8003	10800[a]
13	6783	6783	6783	6783	6783	6877	6783	6783	6877	7075	7075	6783	6783	7075	7277	7445
14	4417	4417	4417	4417	4417	4417	4417	4417	4417	4417	6567	4417	4417	4417	6567	6567
15	7072	7072	7072	7072	7072	7072	7072	7072	7072	7202	8430	7072	7072	7202	9732	9889
16	8440	8440	8616	9717	9948	9948	8440	9816	9948	10046	10047	8440	9948	10046	10047[a]	11315[a]
17	9003	9003	10627	10627	10627	11003	9003	10695	11048	11318	12156	9003	10695	12126	12156	13555[a]
18	6760	6911	6911	6911	6911	6911	6911	6911	6911	8782	10058	6911	6911	8959	10073	10073
19	7051	7051	7051	7051	8428	9941	7051	7051	9996	10267	10267	7051	8492	10267	10267[a]	10267[a]
20	9786	9786	9786	11344	11344	11344	9786	11344	11344	12869	13145	9786	11344	12869	15870	15870[a]
21	4644	4644	4644	4644	4644	4644	4644	4644	4644	4644	7031	4644	4644	4644	7031	7234
22	6442	6442	7744	7990	9209	9209	6442	7990	9341	9499	10854	6688	9341	9499	10854	10854
23	9156	9156	9156	10820	10820	10820	9156	10820	10820	12319	12404	9156	10820	12319	12557	15275[a]
24	11976	11976	11976	11976	11976	11976	11976	11976	11976	15482	16856	11976	11976	15482	16856[a]	16856[a]
25	6346	6346	6346	6346	6346	6346	6346	6346	6346	9025	9025	6346	6346	9025	9272	9272[a]
26	6817	6817	6915	6915	7055	8405	6817	6915	8405	9707	9758	6817	7163	9707	9870	10111[a]
27	9541	9541	9541	9541	9541	10607	9541	9547	10613	10613	12047	9541	9562	10659	12047	13834[a]
28	6782	6782	7235	9545	9767	9767	6782	9545	9767	9894	9894	6782	9767	9894	9894	11209[a]
29	6591	6591	6591	6782	6782	6934	6591	6782	6934	6934	6934	6672	6782	7060	9629	9650
30	6919	6919	7101	7152	7180	8547	6919	7180	8547	9912	9912	6919	7180	9912	9955	10102[a]

Notes: [a] indicate that those instances have no feasible solution since the robust quantity of a supplier exceeds the capacity of vehicles. Therefore, we assume that supply quantity is equal to the capacity of the vehicle to obtain feasible solutions.

Table C13
Computational results of the ALNS algorithm for solving VRPCD-DU in Set 2 instances

Set 2	Combination															
Instance	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	6426	6426	6426	6426	6426	6426	6426	6426	6426	7396	7396	6426	6426	7396	7356	7429
2	6519	6519	6519	6519	6519	7380	6519	6519	7380	7397	7406	6519	6519	7397	7358	7379
3	6369	6369	6369	6369	6369	6369	6369	6369	6369	7360	7322	6369	6369	7360	7322	7322
4	6259	6259	6259	6259	6259	6259	6259	6259	6259	7249	7241	6259	6259	7249	7269	7249
5	6105	6105	6105	6105	6105	6105	6105	6105	6105	7145	7101	6105	6105	7145	7101	7110
6	5617	5617	5617	5617	5617	5617	5617	5617	5617	6702	6701	5617	5617	6702	6697	6687
7	6615	6615	6615	6615	6615	6629	6615	6615	6629	7235	7261	6615	6615	7235	7281	7275
8	6421	6421	6421	6421	6421	6420	6421	6421	6420	7066	7058	6421	6421	7066	7078	7052
9	6469	6469	6469	6469	6469	6451	6469	6469	6451	6469	7455	6469	6469	6469	7527	7527
10	6541	6541	6541	6541	6541	6491	6541	6541	6491	7241	7222	6541	6541	7241	7222	7222
11	6290	6290	6290	6290	6290	6302	6290	6290	6302	6995	7002	6290	6290	7003	7019	7018
12	6611	6611	6611	6611	6611	6635	6611	6611	6635	7412	7459	6611	6611	7459	7412	7412
13	6575	6575	6575	6575	6575	6646	6575	6575	6646	7761	7729	6575	6575	7751	7751	7751
14	6561	6561	6561	6561	6561	6630	6561	6561	6630	6616	7635	6561	6561	6616	7599	7613
15	6655	6655	6655	6655	6655	6535	6655	6655	6535	7309	7334	6655	6655	7329	7334	7332
16	6603	6603	6603	6603	7613	7624	6603	6603	7624	7626	7624	6603	7613	7638	7635	7613
17	6559	6559	6559	6559	6499	6525	6559	6559	6525	6572	7503	6559	6499	6575	7469	7466
18	6532	6532	6532	6532	6613	6558	6532	6532	6558	7488	7480	6532	6613	7488	7488	7488
19	6562	6562	6562	6562	6566	6482	6562	6562	6482	7280	7269	6562	6566	7256	7296	7272
20	6369	6369	6369	6369	6377	6398	6369	6369	6398	6369	7173	6369	6377	6327	7189	7114
21	6549	6549	6549	6549	6549	6572	6549	6549	6572	7275	7244	6549	6549	7254	7270	7275
22	6511	6511	6511	6511	6571	6528	6511	6511	6528	6549	7369	6511	6571	6510	7370	7401
23	6446	6446	6446	6446	6433	6433	6446	6446	6433	7444	7444	6446	6433	7444	7440	7431
24	6539	6539	6539	6539	6539	6547	6539	6539	6547	6526	7398	6539	6539	6536	7364	7382
25	6490	6490	6490	6490	6490	6490	6490	6490	6490	7360	7360	6490	6490	7360	7360	7360
26	6696	6696	6696	6696	6750	6731	6696	6696	6731	7532	7560	6696	6750	7532	7532	7532
27	6843	6843	6843	6843	6733	6733	6843	6843	6733	7574	7611	6843	6733	7594	7649	7632
28	7132	7132	7132	7132	7124	7085	7132	7132	7085	7820	7811	7132	7124	7811	7820	7745
29	6550	6550	6550	6550	6550	6550	6550	6550	6550	7556	7613	6550	6550	7589	7576	7590
30	6510	6510	6510	6510	6518	6548	6510	6510	6548	6585	7341	6510	6518	6494	7341	7389

Table C14
Computational results of the ALNS algorithm for solving VRPCD-DU in Set 3 instances

Set 3	Combination															
Instance	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	15099	15099	15236	15067	15126	14953	15099	15067	14953	15344	15408	15099	15126	15344	15275	15505
2	13304	13304	13562	13773	13877	13553	13304	13773	13553	13930	13825	13304	13877	13930	14218	13966
3	13242	13242	13150	13259	13253	13059	13242	13259	13059	14018	13820	13242	13221	13756	14221	13675
4	14124	14124	14339	14304	14552	14412	14124	14304	14489	14697	14125	14124	14476	14449	14593	14604
5	13801	13801	13721	14411	14635	14529	13801	14411	14394	14495	14571	13801	14467	14398	14450	14656
6	14375	14375	14795	14410	14500	14534	14375	14410	14577	14495	14563	14375	14641	14526	14342	14828
7	15450	15450	15516	15650	15597	15906	15450	15650	15265	15609	15686	15450	15460	15912	15590	15444
8	13715	13715	14032	13937	14433	14178	13715	13937	14551	14444	14727	13715	14454	14656	14151	14737
9	13618	13618	13587	13644	13416	13496	13618	13644	13625	13664	13894	13618	13811	13760	13946	13864
10	15143	15143	15551	15161	15474	15301	15143	15161	15345	15286	15666	15143	15420	15338	15603	15481
11	14278	14278	13879	14155	14322	14059	14278	14155	14165	13950	14443	14278	14365	14188	14533	14511
12	12722	12722	12550	12828	12802	12548	12722	12828	12833	13042	12959	12722	12677	12737	12897	13504
13	12559	12559	12459	12746	12648	12612	12559	12746	12736	13069	13258	12559	12259	12908	13200	13237
14	13177	13177	13060	12979	12971	12945	13177	12979	12976	12993	13715	13177	13097	13202	14062	14050
15	14872	14872	14909	14696	14791	14962	14872	14696	14750	14675	14771	14872	14753	14887	14859	15187
16	13888	13888	13572	14105	13486	13943	13888	14105	13583	13880	13925	13888	14097	13838	13979	13705
17	13959	13959	14013	13984	13797	13811	13959	13984	13852	13587	13830	13959	13766	13927	14390	13941
18	13740	13740	14124	13630	13686	13746	13740	13630	13979	13784	14288	13740	14014	13937	14383	14279
19	13794	13794	14283	14565	14067	14294	13794	14607	14217	14203	14202	14425	14432	14457	14589	14723
20	13069	13069	13476	13618	13447	13446	13069	13493	13375	13246	13785	13608	13248	13402	14030	14270
21	13948	13948	13900	13847	13637	13910	13948	14017	13844	14076	14494	13700	13671	14093	14658	14539
22	13629	13629	13419	13476	13450	13873	13629	13337	14205	13755	14022	13740	13425	14052	13797	13886
23	13678	13678	13796	14482	14919	14782	13678	14410	14811	14681	14936	14057	14897	14563	14775	15453
24	13287	13287	12813	13432	13397	13404	13287	13459	13112	13393	13600	12922	13610	13595	13536	14029
25	13624	13624	14202	14025	14421	13712	13624	13786	13464	14162	13968	14100	13737	14300	14196	14129
26	13468	13468	13469	13466	13980	13919	13468	13945	14130	13864	13913	13703	13853	13940	13936	13861
27	13783	13783	14183	14225	14217	14312	14009	13989	14260	13974	14072	13920	14033	14107	14346	14686
28	13771	13771	13834	13648	13683	14026	13722	13559	14097	14221	14250	13910	14097	14155	14346	14060
29	14004	14004	13958	13988	13845	14533	13975	14074	14184	14201	14511	14143	14060	14077	14308	14801
30	13664	13664	13795	13788	13706	13823	13694	13658	13611	13563	14012	13778	13710	13810	14126	14550

Table C15
Cost ratio on the optimal solutions of instance 10-4 in Set 1

ε, Γ	Total cost					Cost ratio				
	0	0.25	0.5	0.75	1	0	0.25	0.5	0.75	1
0	7229	7229	7229	7229	7229	1	1	1	1	1
0.2	7229	7229	7255	7309	7309	1	1	1.004	1.011	1.011
0.4	7229	7255	7309	7412	7739	1	1.004	1.011	1.025	1.071
0.6	7229	7309	7412	8710	10148	1	1.011	1.025	1.205	1.404

Table C16
Optimal routing plan obtained by the VRPCD model of instance 10-4 in Set 1

Route	Sequence						Nominal loading	Worst-case loading	Unmet quantity
Route 1	0	C3	C1	C6	C2	0	66	85.8	15.8
Route 2	0	C5	C4	0			28	36.4	-
Route 3	0	S2	S4	S3	0		56	72.8	2.8
Route 4	0	S1	0				38	49.4	-

Table C17
Utilization of vehicle capacity in VRPCD solutions under the worst-case realization of uncertainty levels with the original fleet size

Combi.	ε	Γ	Set 1						Set 2						Set 3					
			Customer			Supplier			Customer			Supplier			Customer			Supplier		
			Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max	Average	Min	Max
0	0	0	0.77	0.34	1.00	0.73	0.20	1.00	0.39	0.30	0.50	0.79	0.77	0.87	0.40	0.22	0.69	0.72	0.28	0.98
1	0.2	0	0.78	0.37	**1.04**	0.73	0.20	1.00	0.41	0.31	0.52	0.79	0.77	0.87	0.43	0.23	0.74	0.72	0.28	0.98
2	0.2	0.25	0.81	0.38	**1.09**	0.77	0.21	**1.05**	0.43	0.33	0.54	0.83	0.80	0.92	0.45	0.24	0.76	0.75	0.29	**1.03**
3	0.2	0.5	0.85	0.39	**1.12**	0.80	0.22	**1.10**	0.45	0.34	0.56	0.87	0.84	0.96	0.46	0.25	0.78	0.79	0.31	**1.08**
4	0.2	0.75	0.89	0.40	**1.16**	0.84	0.23	**1.15**	0.46	0.35	0.58	0.91	0.88	**1.00**	0.47	0.26	0.81	0.83	0.32	**1.13**
5	0.2	1	0.92	0.41	**1.20**	0.88	0.24	**1.20**	0.47	0.36	0.60	0.95	0.92	**1.05**	0.48	0.26	0.83	0.86	0.34	**1.18**
6	0.4	0	0.79	0.39	**1.09**	0.73	0.20	1.00	0.43	0.33	0.54	0.79	0.77	0.87	0.46	0.24	0.80	0.72	0.28	0.98
7	0.4	0.25	0.86	0.42	**1.20**	0.80	0.22	**1.10**	0.46	0.35	0.58	0.87	0.84	0.96	0.49	0.25	0.85	0.79	0.31	**1.08**
8	0.4	0.5	0.94	0.44	**1.26**	0.88	0.24	**1.20**	0.50	0.38	0.62	0.95	0.92	**1.05**	0.52	0.27	0.89	0.86	0.34	**1.18**
9	0.4	0.75	**1.00**	0.46	**1.32**	0.95	0.26	**1.30**	0.53	0.40	0.66	**1.03**	1.00	**1.14**	0.54	0.29	0.92	0.93	0.36	**1.27**
10	0.4	1	**1.07**	0.48	**1.40**	**1.02**	0.28	**1.40**	0.55	0.42	0.70	**1.11**	**1.07**	**1.22**	0.56	0.31	0.97	**1.00**	0.39	**1.37**
11	0.6	0	0.80	0.39	**1.14**	0.73	0.20	1.00	0.45	0.34	0.56	0.79	0.77	0.87	0.49	0.24	0.87	0.72	0.28	0.98
12	0.6	0.25	0.91	0.44	**1.30**	0.84	0.23	**1.15**	0.50	0.38	0.62	0.91	0.88	**1.00**	0.53	0.27	0.94	0.83	0.32	**1.13**
13	0.6	0.5	**1.02**	0.49	**1.39**	0.95	0.26	**1.30**	0.55	0.42	0.68	**1.03**	1.00	**1.14**	0.57	0.30	0.99	0.93	0.36	**1.27**
14	0.6	0.75	**1.12**	0.52	**1.49**	**1.06**	0.29	**1.45**	0.59	0.45	0.74	**1.15**	**1.11**	**1.27**	0.61	0.33	**1.04**	**1.04**	0.41	**1.42**
15	0.6	1	**1.23**	0.55	**1.60**	**1.17**	0.32	**1.60**	0.63	0.48	0.80	**1.26**	**1.23**	**1.40**	0.64	0.35	**1.11**	**1.15**	0.45	**1.57**

Notes: Bold values emphasize that the average, minimum, and maximum loadings of routes exceed the capacity of the vehicle under the uncertainty level (ε, Γ). Combination 0 represent the deterministic VRPCD with $(\varepsilon, \Gamma) = (0, 0)$.

Table C18
Utilization of vehicle capacity in VRPCD solutions under the worst-case realization of uncertainty levels with varying fleet sizes

Combination	Set 2										Set 3									
	C135		C120		C105		C90		C75		C135		C120		C105		C90		C75	
	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
0	0.44	0.56	0.49	0.63	0.56	0.71	0.66	0.83	0.79	1.00	0.45	0.77	0.50	0.87	0.57	0.99	0.67	**1.16**	0.80	**1.39**
1	0.46	0.58	0.52	0.65	0.59	0.74	0.69	0.86	0.82	**1.04**	0.48	0.82	0.54	0.92	0.61	**1.05**	0.72	**1.23**	0.86	**1.47**
2	0.48	0.60	0.54	0.68	0.61	0.77	0.72	0.90	0.86	**1.08**	0.50	0.85	0.56	0.95	0.64	**1.09**	0.74	**1.27**	0.89	**1.53**
3	0.50	0.62	0.56	0.70	0.64	0.80	0.74	0.94	0.89	**1.12**	0.51	0.87	0.57	0.98	0.66	**1.11**	0.77	**1.30**	0.92	**1.56**
4	0.51	0.65	0.58	0.73	0.66	0.83	0.77	0.97	0.92	**1.16**	0.52	0.90	0.59	**1.01**	0.67	**1.15**	0.79	**1.35**	0.94	**1.62**
5	0.53	0.67	0.59	0.75	0.68	0.86	0.79	1.00	0.95	**1.20**	0.54	0.92	0.60	**1.04**	0.69	**1.19**	0.80	**1.39**	0.96	**1.66**
6	0.48	0.60	0.54	0.67	0.61	0.77	0.72	0.90	0.86	**1.07**	0.51	0.89	0.57	**1.00**	0.65	**1.14**	0.76	**1.34**	0.92	**1.60**
7	0.52	0.65	0.58	0.73	0.66	0.83	0.77	0.97	0.93	**1.16**	0.55	0.95	0.61	**1.07**	0.70	**1.22**	0.82	**1.42**	0.98	**1.71**
8	0.55	0.69	0.62	0.78	0.71	0.89	0.83	**1.04**	0.99	**1.25**	0.57	0.99	0.65	**1.11**	0.74	**1.27**	0.86	**1.48**	**1.03**	**1.77**
9	0.58	0.73	0.66	0.83	0.75	0.94	0.88	**1.10**	**1.05**	**1.32**	0.60	**1.02**	0.68	**1.15**	0.77	**1.32**	0.90	**1.54**	**1.08**	**1.84**
10	0.61	0.78	0.69	0.88	0.79	1.00	0.92	**1.17**	**1.11**	**1.40**	0.63	**1.08**	0.70	**1.21**	0.80	**1.39**	0.94	**1.62**	**1.13**	**1.94**
11	0.50	0.62	0.56	0.70	0.64	0.79	0.75	0.93	0.89	**1.11**	0.54	0.96	0.61	**1.08**	0.69	**1.24**	0.81	**1.44**	0.97	**1.73**
12	0.56	0.69	0.62	0.78	0.71	0.89	0.83	**1.04**	1.00	**1.25**	0.59	**1.05**	0.67	**1.18**	0.76	**1.35**	0.89	**1.57**	**1.07**	**1.89**
13	0.61	0.76	0.68	0.86	0.78	0.98	0.91	**1.14**	**1.09**	**1.37**	0.64	**1.10**	0.72	**1.24**	0.82	**1.42**	0.96	**1.66**	**1.15**	**1.99**
14	0.66	0.82	0.74	0.93	0.84	**1.06**	0.98	**1.24**	**1.18**	**1.48**	0.68	**1.16**	0.76	**1.30**	0.87	**1.49**	**1.02**	**1.74**	**1.22**	**2.08**
15	0.70	0.89	0.79	1.00	0.90	**1.14**	**1.05**	**1.33**	**1.26**	**1.60**	0.71	**1.23**	0.80	**1.39**	0.92	**1.58**	**1.07**	**1.85**	**1.29**	**2.22**

Note: Bold values emphasize that the average and maximum loadings of routes exceed the capacity of the vehicle under the uncertainty level (ε, Γ).

Table C19
Empirical probability of loading violations and the price of robustness (Unit: %)

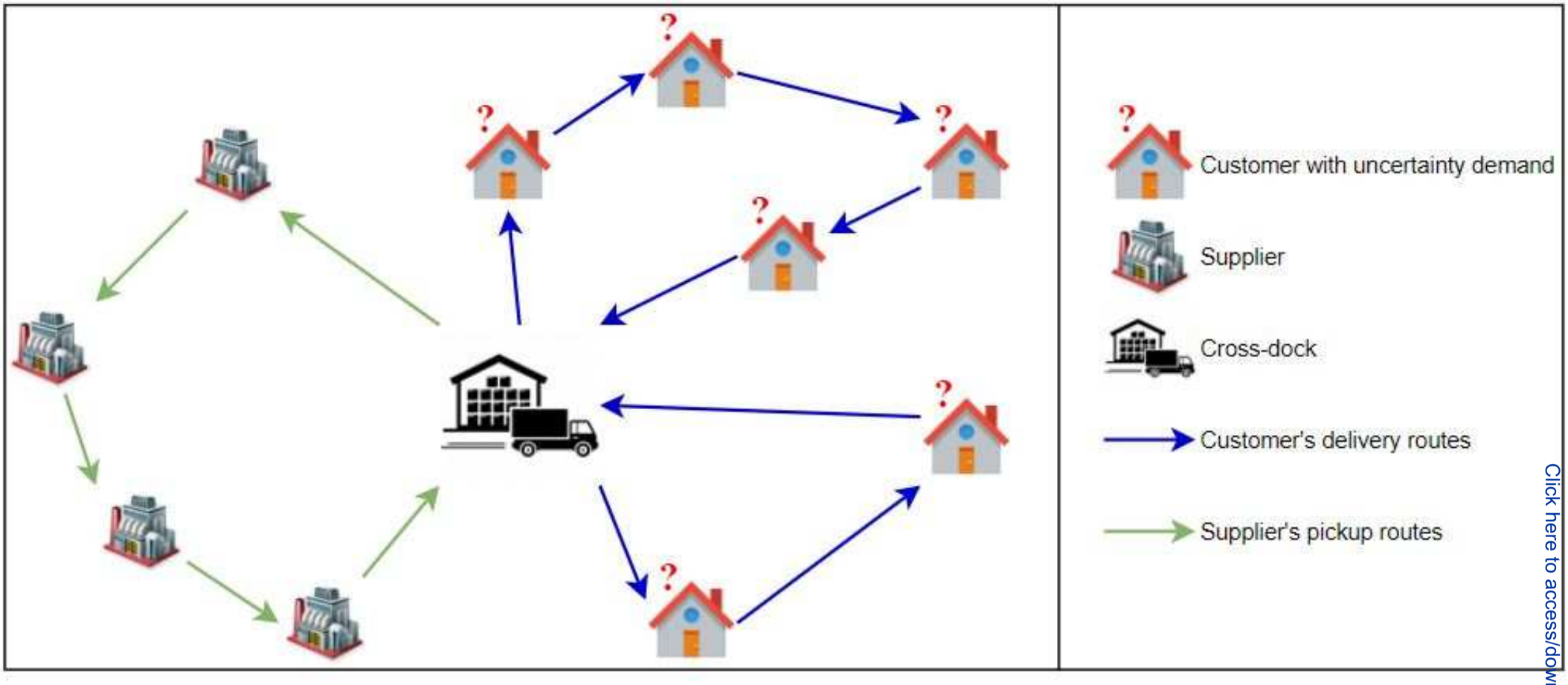
	Combination	Set 1				Set 2				Set 3			
		$\mathscr{P}$	$\mathscr{P}_C$	$\mathscr{P}_S$	PoR	$\mathscr{P}$	$\mathscr{P}_C$	$\mathscr{P}_S$	PoR	$\mathscr{P}$	$\mathscr{P}_C$	$\mathscr{P}_S$	PoR
"Cluster" instances	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0.07	0.07	0	0.79	0	0	0	**-0.10**	0	0	0	0.40
	2	8.54	5.45	3.09	3.82	0	0	0	**-0.05**	0.99	0.75	0.24	0.62
	3	26.85	15.04	13.59	8.78	0	0	0	0.16	12.07	5.14	6.92	1.02
	4	43.72	24.50	24.72	10.34	0	0	0	0.54	19.76	6.52	14.60	0.95
	5	56.54	33.85	34.22	13.74	0	0	0	0.79	29.29	6.65	25.65	1.52
	6	0.36	0.36	0	1.39	0	0	0	**-0.05**	0.03	0.03	0	0.41
	7	12.88	8.37	4.95	8.88	0	0	0	**-0.16**	4.88	2.12	2.75	0.34
	8	40.65	25.87	21.82	13.89	0	0	0	0.96	21.56	5.90	18.08	1.37
	9	61.93	44.19	38.02	22.36	1.60	0.13	1.57	10.25	40.43	7.67	37.10	2.38
	10	74.88	58.86	49.92	31.32	10.50	2.89	9.16	14.33	53.77	10.27	50.32	3.33
	11	0.75	0.75	0	1.60	0	0	0	0.09	0.10	0.10	0	0.08
	12	18.55	13.06	6.79	10.63	0	0	0	0.60	7.09	2.64	4.62	1.06
	13	50.97	36.37	28.65	24.27	0.60	0.15	0.51	10.22	33.20	7.18	30.08	2.71
	14	72.36	57.43	48.06	37.68	15.65	5.60	13.20	16.30	58.47	12.99	54.41	3.95
	15	84.67	72.92	65.66	46.61	63.84	21.37	61.73	28.93	75.78	22.44	70.59	5.53
	Average	**34.61**	**24.82**	**21.22**	**14.76**	**5.76**	**1.88**	**5.39**	**5.18**	**22.34**	**5.65**	**19.71**	**1.60**
"Random" instances	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	2.56	2.56	0	1.19	0	0	0	**-0.02**	1.15	1.15	0	0.15
	2	19.03	10.71	8.56	4.64	0	0	0	**-0.02**	11.17	5.72	5.45	0.52
	3	36.29	20.87	19.21	8.82	0	0	0	**-0.12**	19.43	6.63	12.90	0.94
	4	51.03	29.65	29.67	10.41	0	0	0	0.61	32.14	6.67	28.58	1.16
	5	61.40	38.34	37.30	13.74	0.55	0	0.55	1.04	40.49	6.81	37.14	1.52
	6	3.69	3.69	0	1.46	0	0	0	**-0.03**	2.21	2.21	0	0.19
	7	25.46	16.16	12.05	9.35	0	0	0	0.19	16.78	6.41	10.42	0.88
	8	51.03	33.98	29.25	15.52	1.30	0.34	1.14	1.17	42.32	7.30	38.96	1.35
	9	67.62	50.35	42.05	22.91	12.47	4.39	10.60	10.32	56.15	10.28	52.33	2.40
	10	79.39	64.15	54.47	31.32	43.44	14.80	40.25	14.33	69.14	14.02	64.70	3.07
	11	4.57	4.57	0	2.80	0	0	0	0.07	2.89	2.89	0	0.19
	12	32.36	22.18	15.38	12.47	0.26	0.26	0	0.56	26.50	6.68	22.12	1.21
	13	60.88	45.33	35.49	25.14	14.72	6.96	11.04	10.85	57.48	11.14	53.20	2.73
	14	79.54	65.95	55.92	39.00	76.67	26.73	75.71	19.78	78.92	23.40	72.80	4.71
	15	89.23	80.02	72.94	46.61	98.10	49.64	98.03	28.93	91.52	33.95	86.19	5.53
	Average	**41.50**	**30.53**	**25.77**	**15.34**	**15.47**	**6.44**	**14.83**	**5.48**	**34.27**	**9.08**	**30.30**	**1.66**

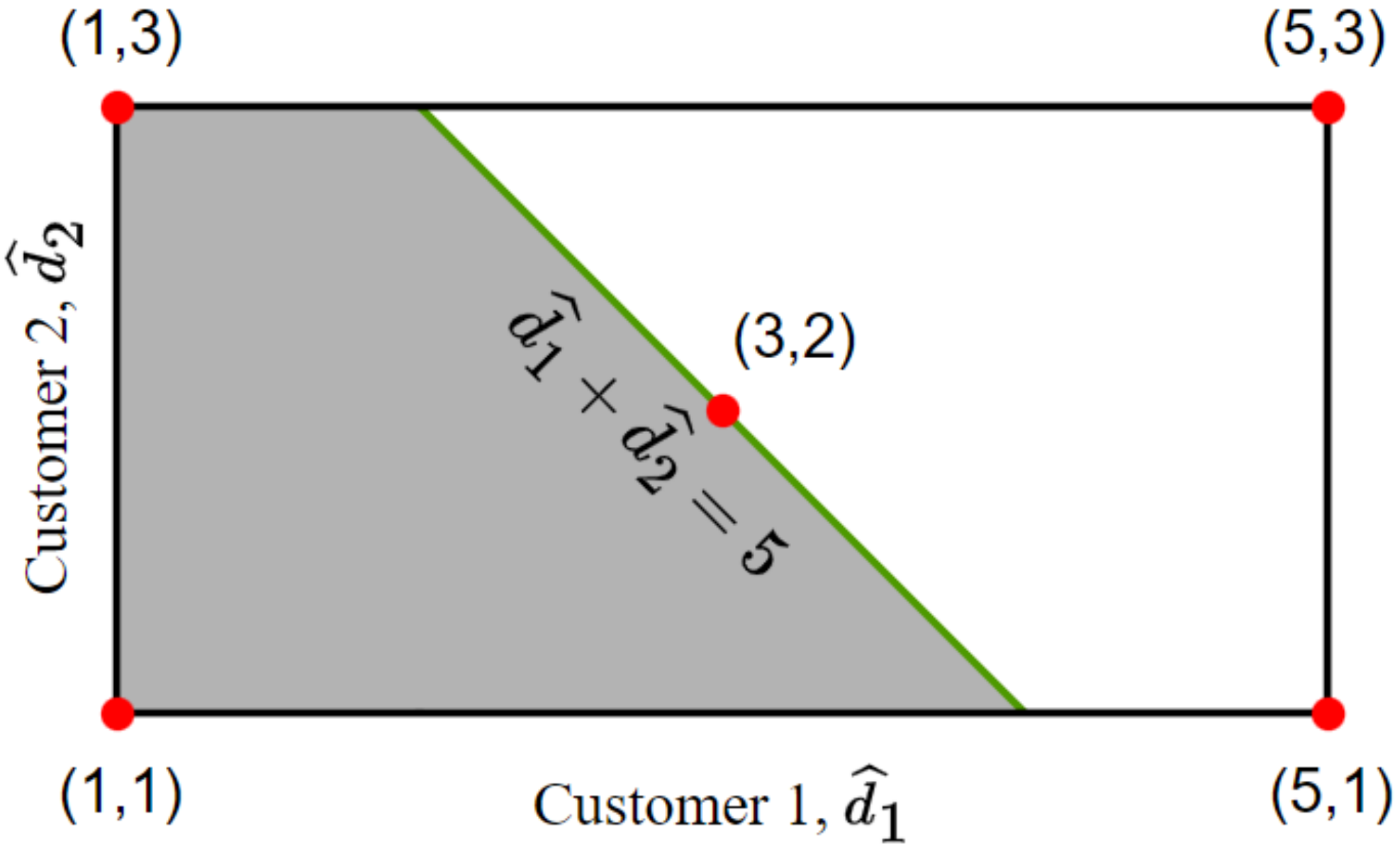
Note: Bold PoR values probably are negligible, where a small change in average costs can be ignored since solutions from ALNS are suboptimal.

Click here to access/download

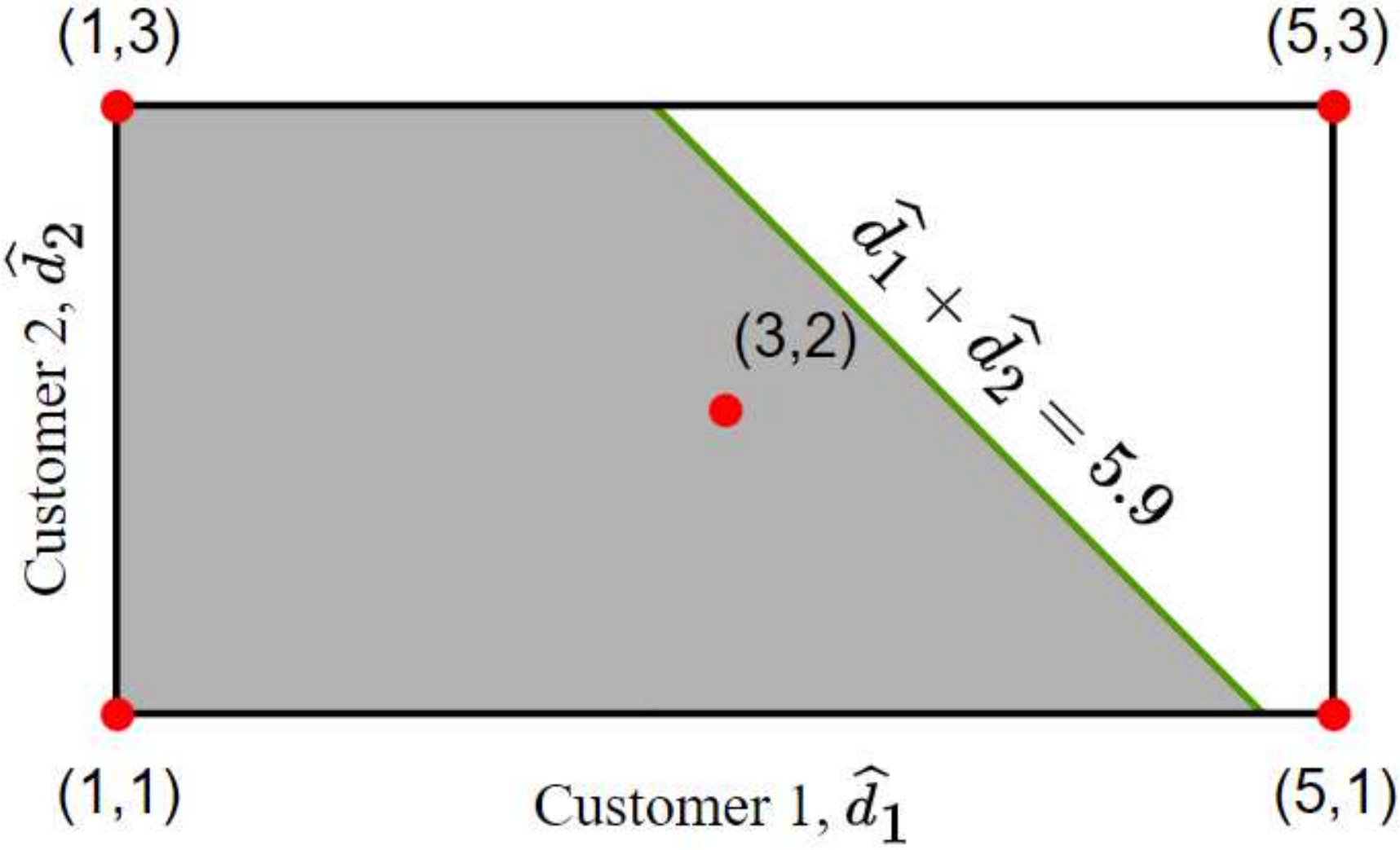
LaTeX Source File

VRPCD-DU_R3_2023.02.06.zip

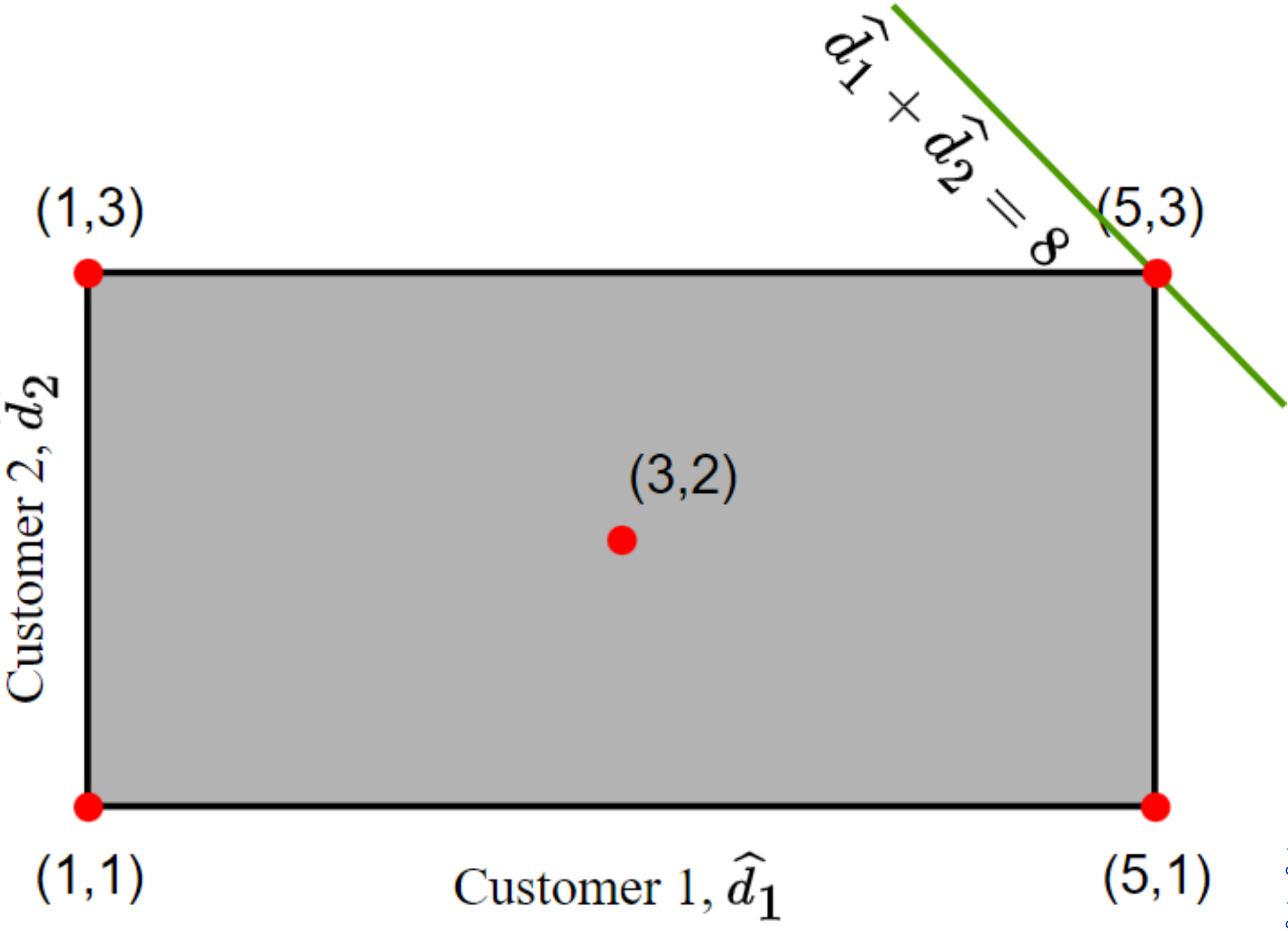




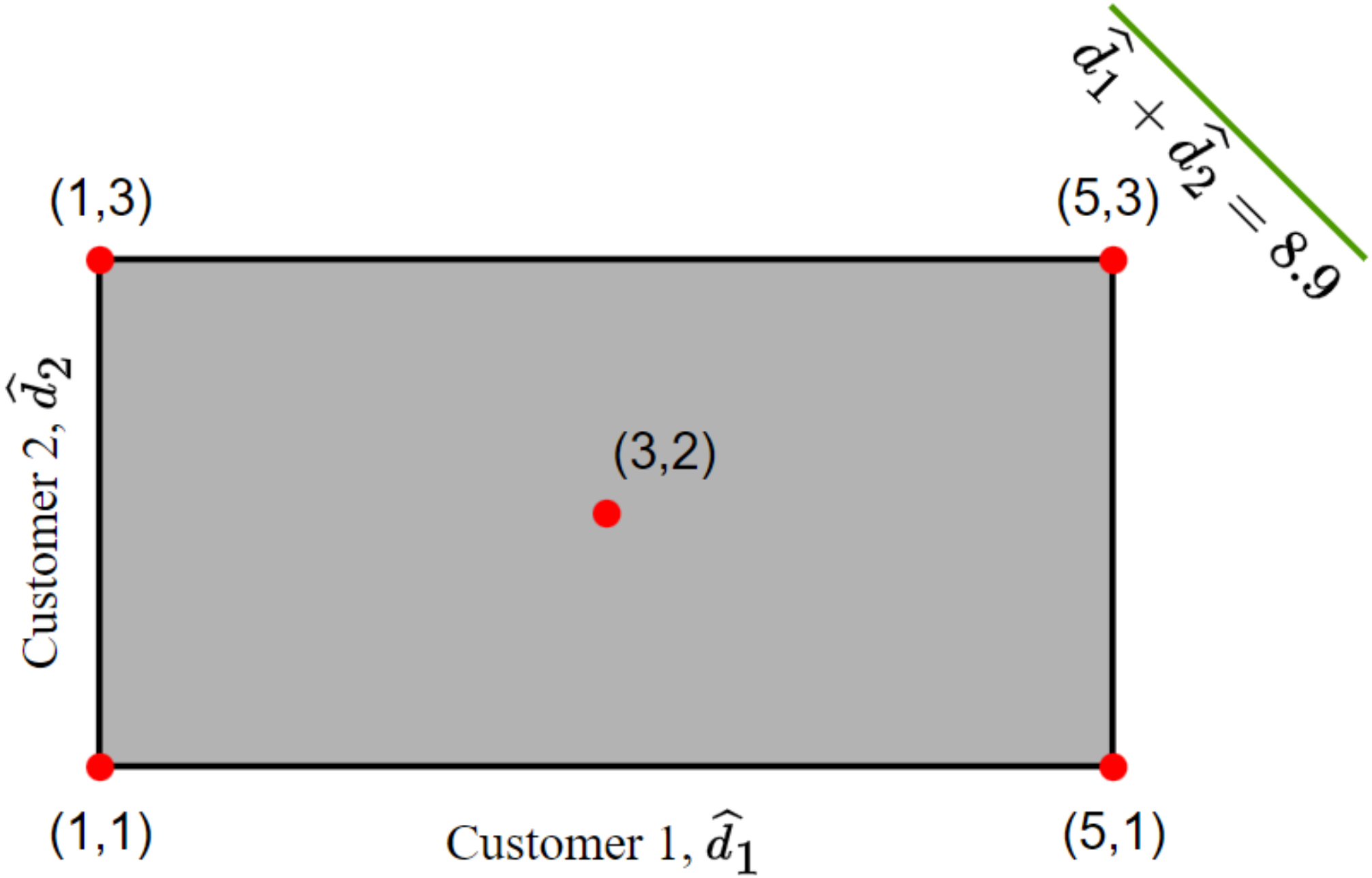
(a) Case 1: $\Gamma = 0$



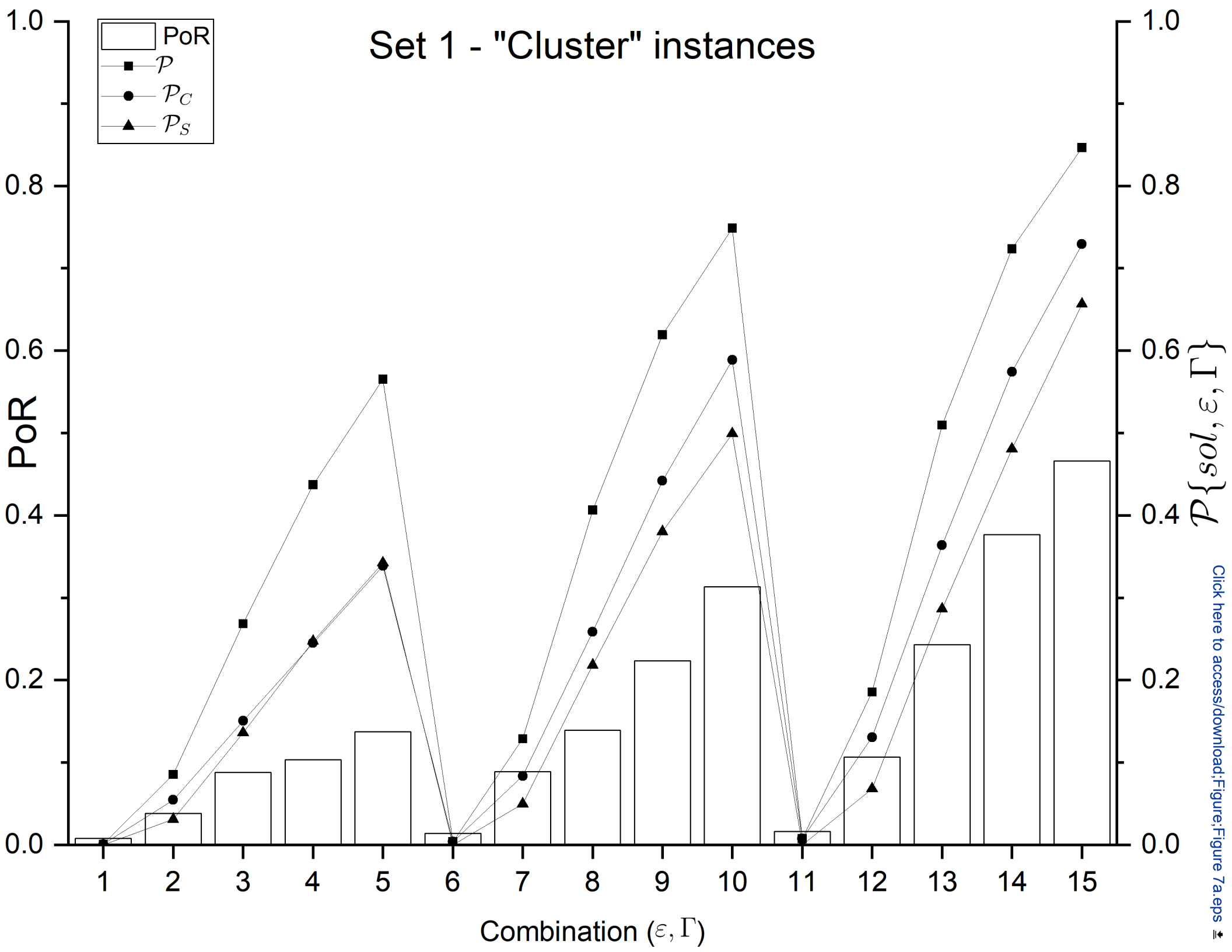
(b) Case 2: $0 < \Gamma < 1$

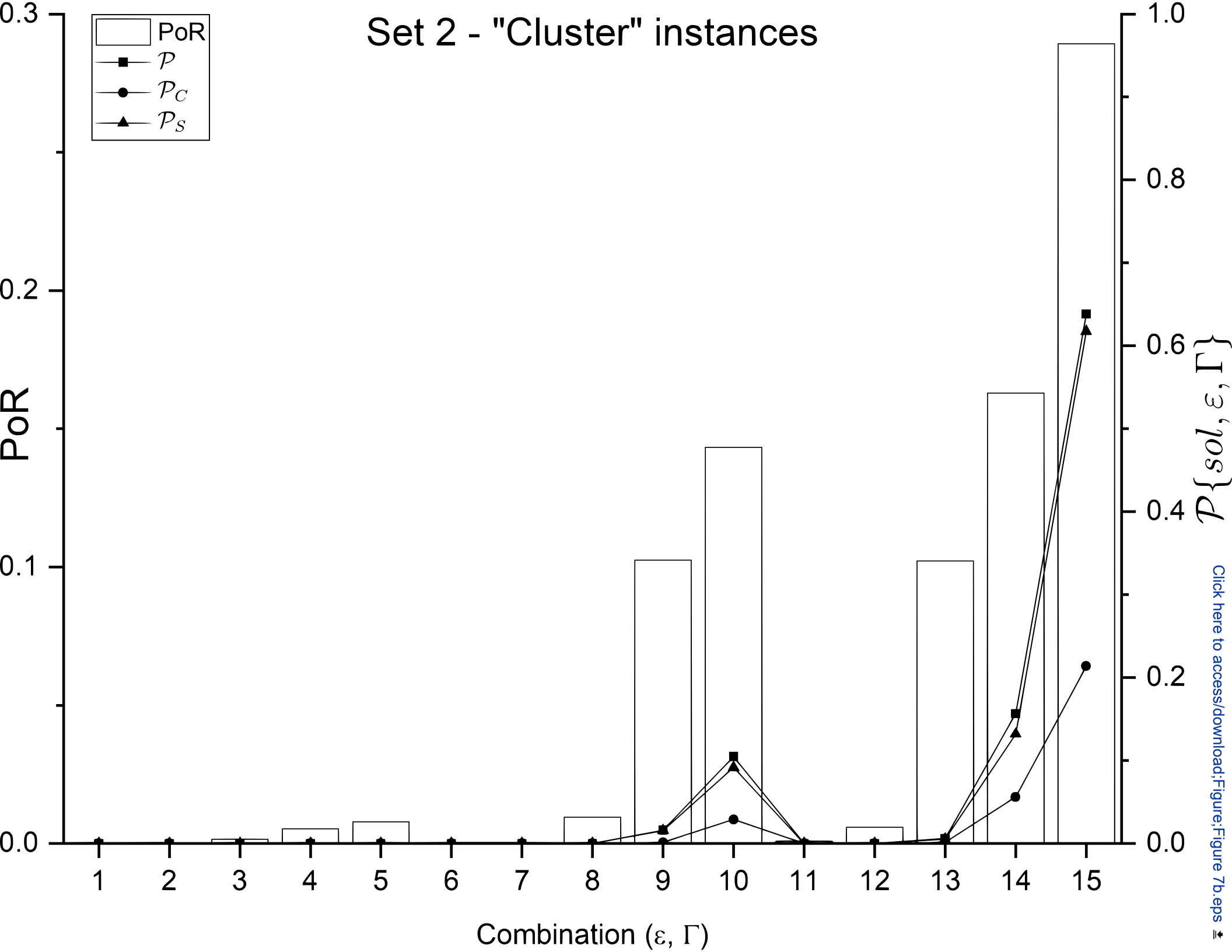


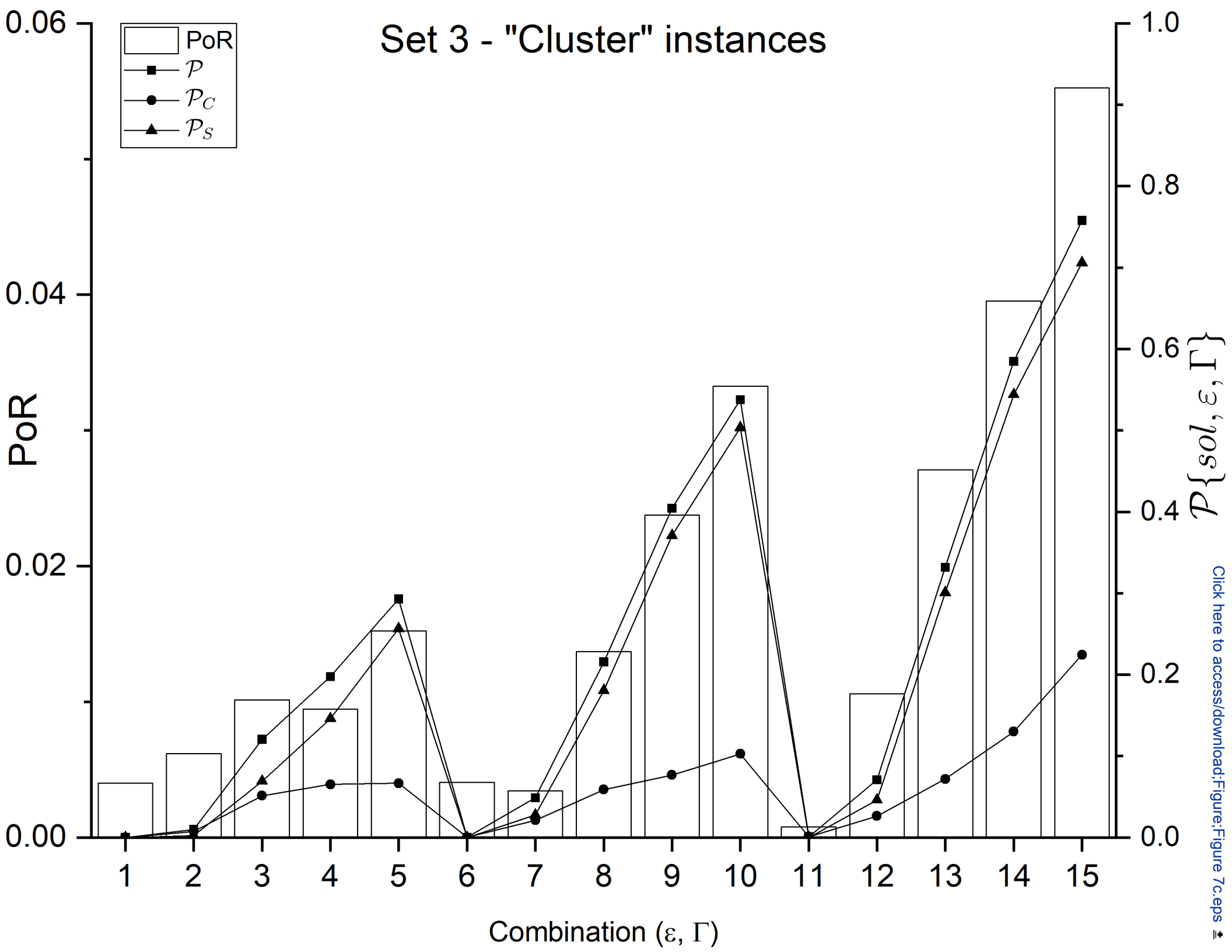
(c) Case 3: $\Gamma = 1$

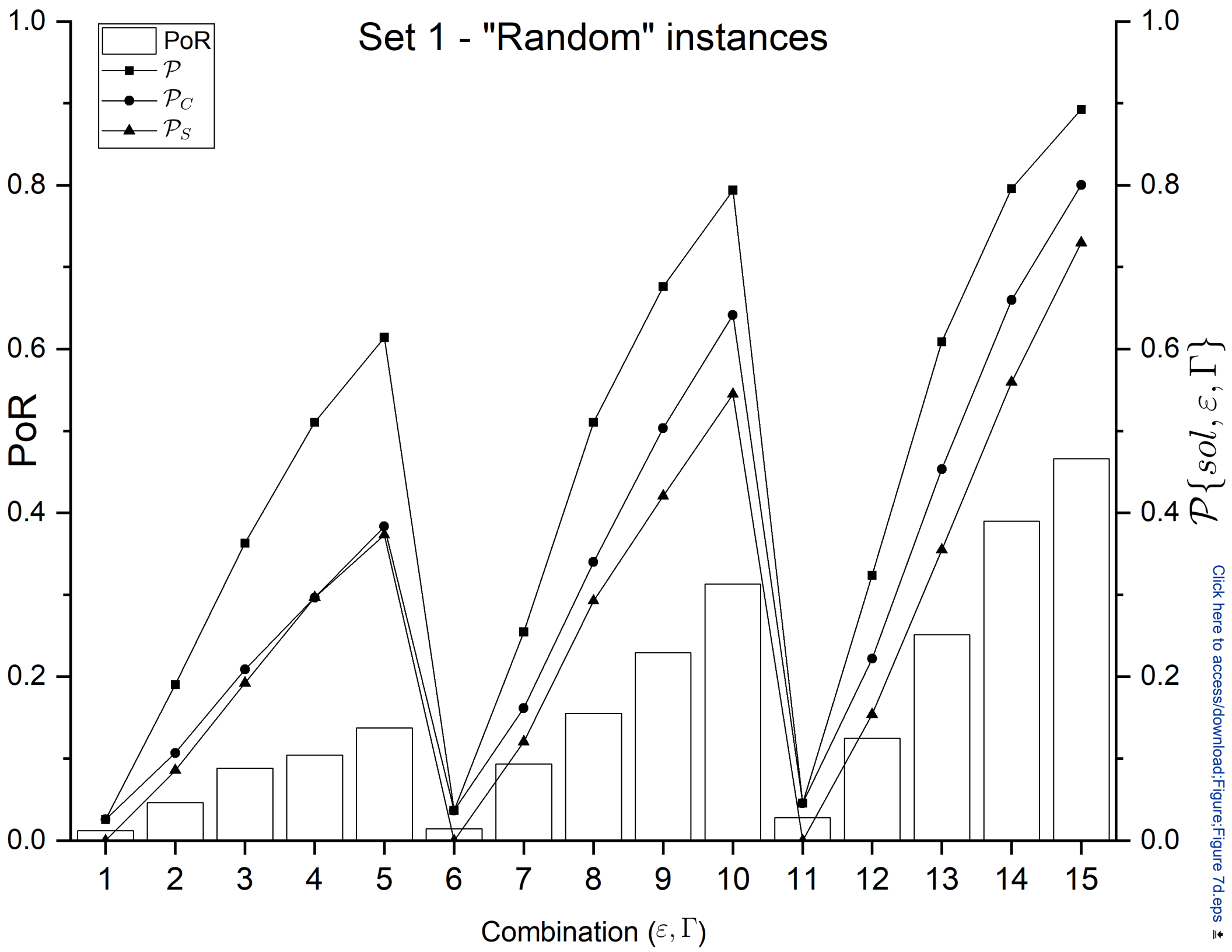


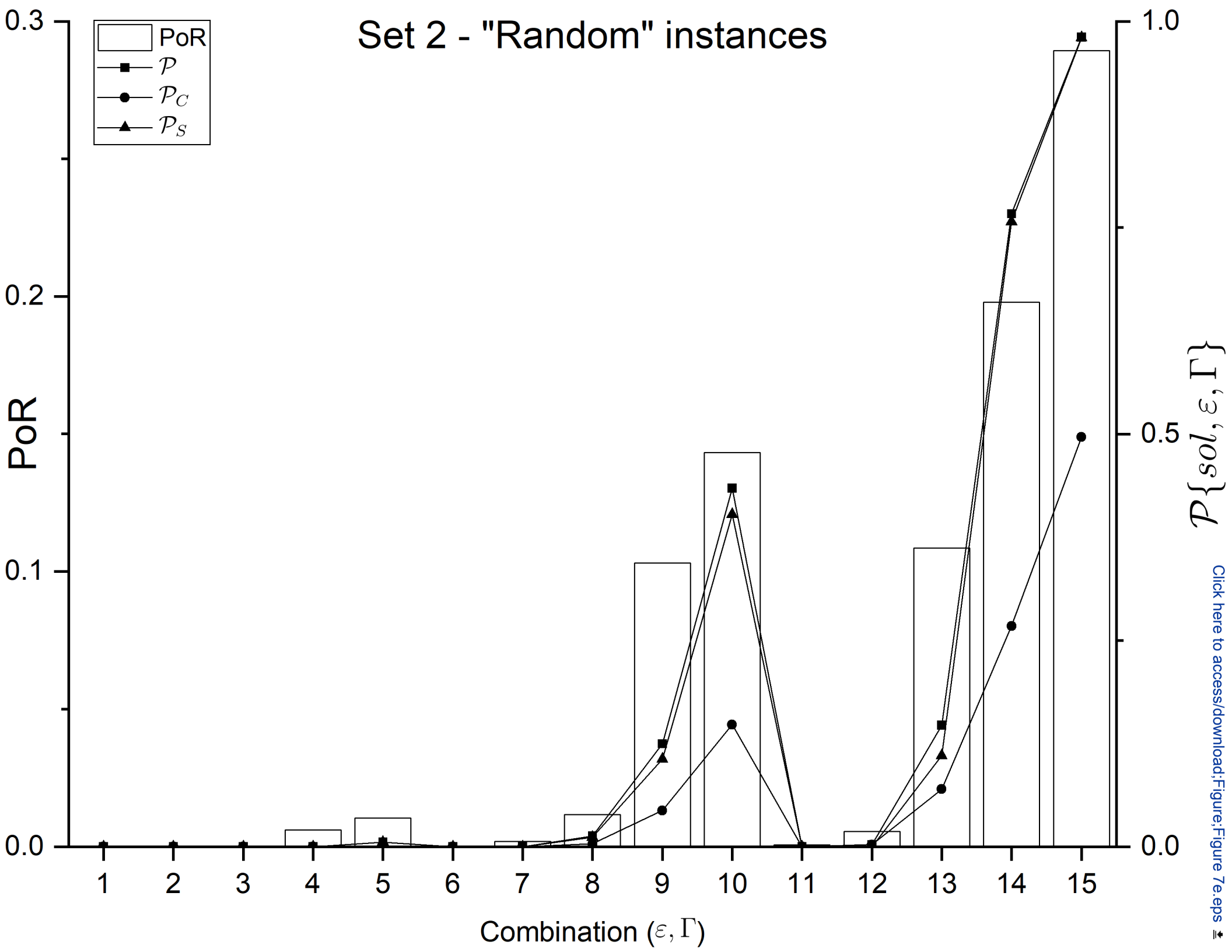
(d) Case 4: $\Gamma > 1$

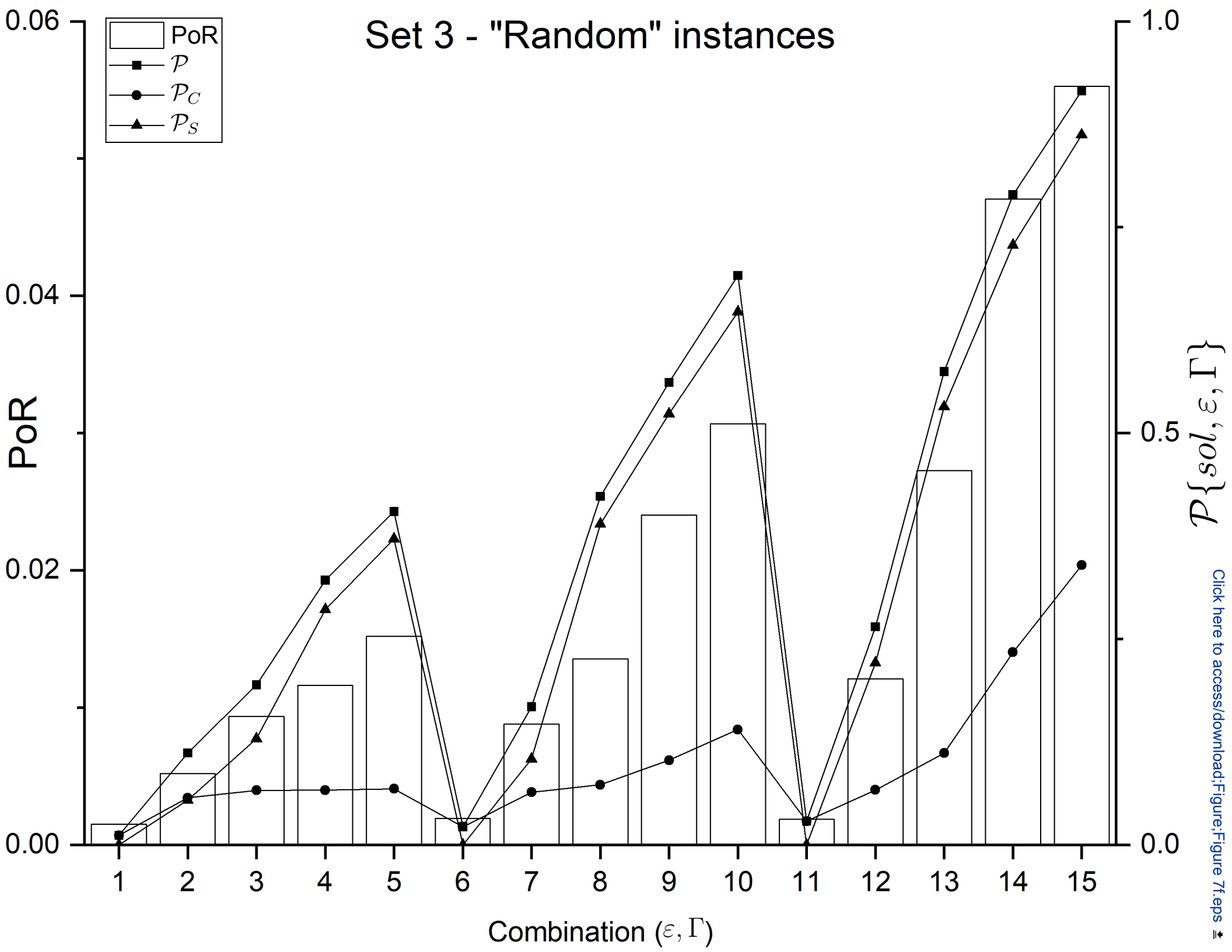


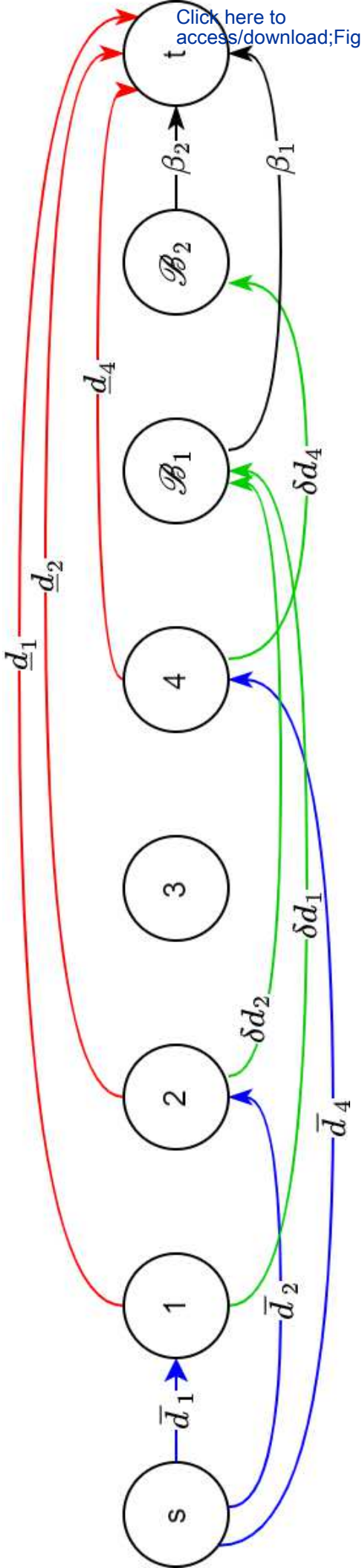


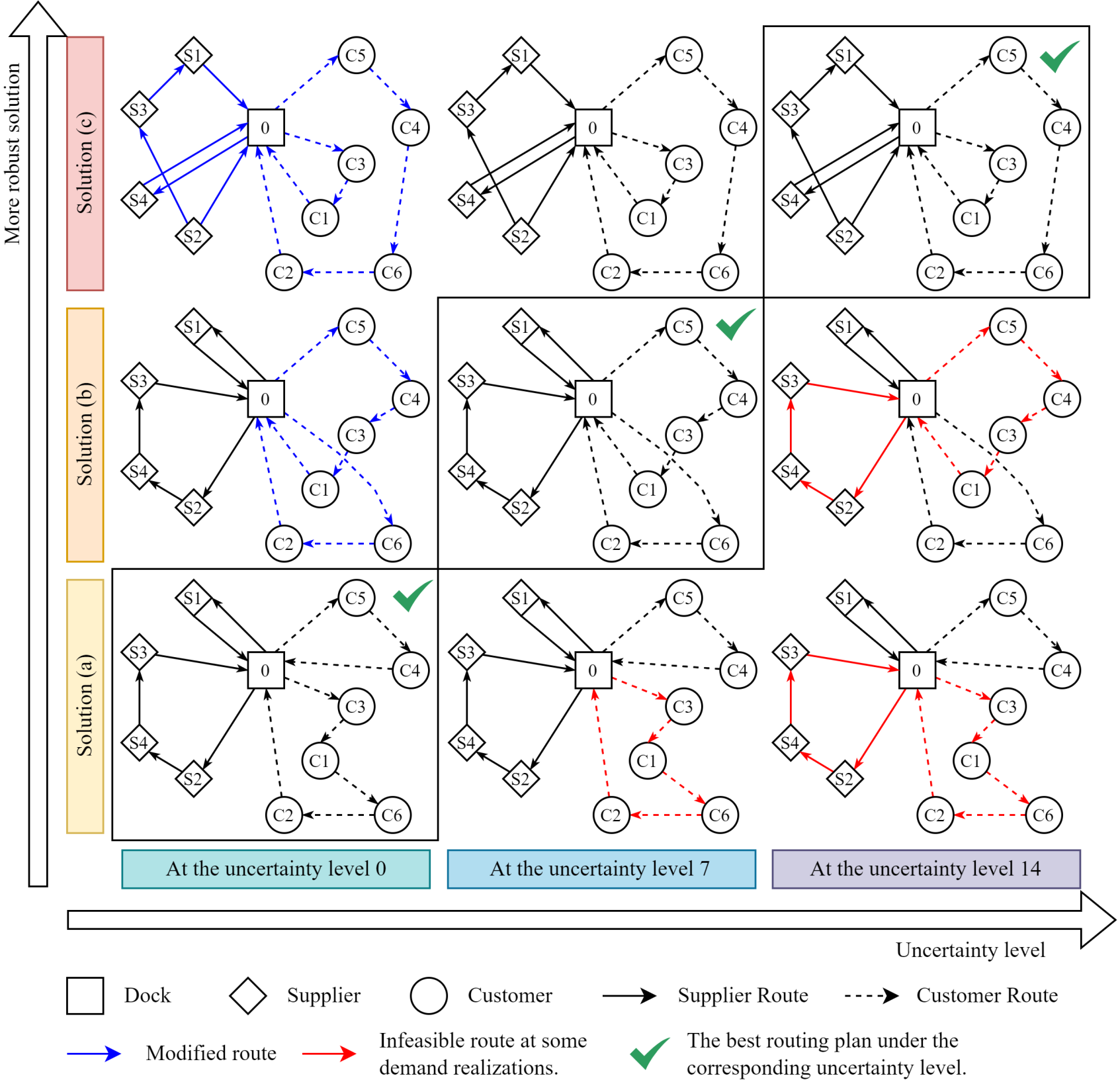


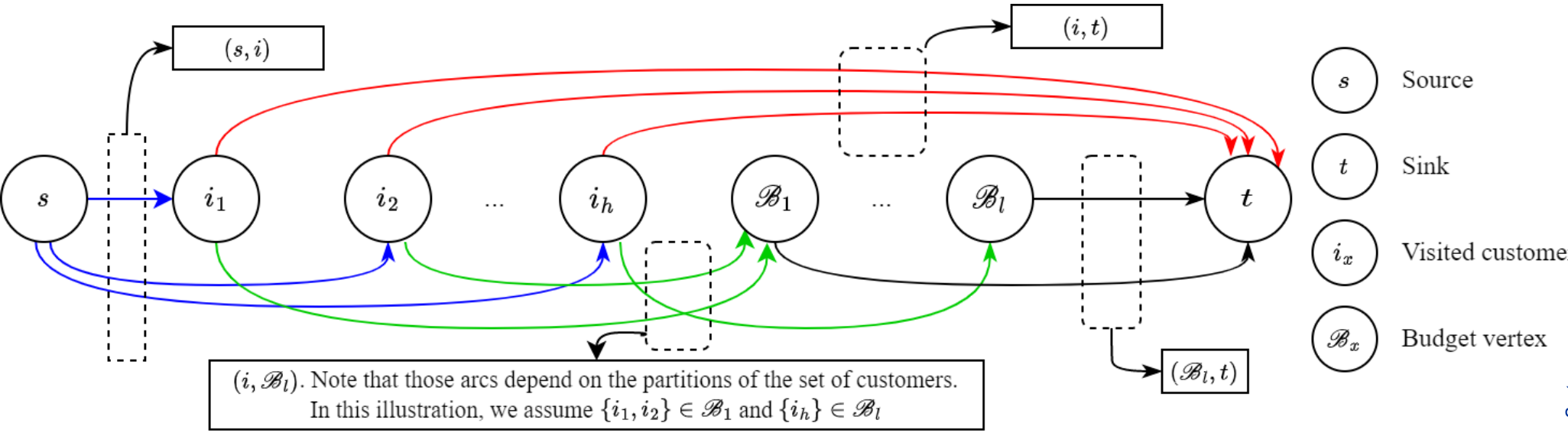












Name	CRediT statements
Vincent F. Yu	Conceptualization, Methodology, Supervision, Resources, Writing - Review & Editing
Pham Tuan Anh	Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Writing - Original Draft
Roberto Baldacci	Formal Analysis, Writing - Review & Editing

Click here to access/download

Supplementary Material

VRPCD_DU_R1_SupMaterials_2022.09.01.xlsx