Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

Junko Hosoda

Stephen J. Maher

Yuji Shinano

Jonas Christoffer Villumsen

A parallel branch-and-bound heuristic for the integrated long-haul and local vehicle routing problem on an adaptive transportation network

ZIB Report 23-02 (January 2023)

Zuse Institute Berlin

Takustr. 7

14195 Berlin

Germany

Telephone: +49 30 84185-0

Telefax: +49 30 84185-125

E-mail: bibliothek@zib.de

URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064

ZIB-Report (Internet) ISSN 2192-7782

A parallel branch-and-bound heuristic for the integrated long-haul and local vehicle routing problem on an adaptive transportation network

Junko Hosoda[1], Stephen J. Maher[1,2,3], Yuji Shinano[4], and Jonas Christoffer Villumsen[5]

[1]Controls and Robotics Innovation Center, Hitachi Ltd., Kanagawa, Japan

[2]College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, United Kingdom

[3]Quantagonia GmbH, Bad Homburg, Germany

[4]Department of Applied Algorithmic Intelligence Methods (A²IM), Zuse Institute Berlin, Berlin, Germany

[5]Instrumentation Innovation Center, Hitachi Ltd., Tokyo, Japan

Abstract

Consolidation of commodities and coordination of vehicle routes are fundamental features of supply chain management problems. While locations for consolidation and coordination are typically known a priori, in adaptive transportation networks this is not the case. The identification of such consolidation locations forms part of the decision making process. Supply chain management problems integrating the designation of consolidation locations with the coordination of long haul and local vehicle routing is not only challenging to solve, but also very difficult to formulate mathematically. In this paper, the first mathematical model integrating location clustering with long haul and local vehicle routing is proposed. This mathematical formulation is used to develop algorithms to find high quality solutions. A novel parallel framework is developed that combines exact and heuristic methods to improve the search for high quality solutions and provide valid bounds. The results demonstrate that using exact methods to guide heuristic search is an effective approach to find high quality solutions for difficult supply chain management problems.

[1]stephen@sjmsolutions.co.uk

Key words: supply chain management, vehicle routing, location, parallel computing, synchronisation

1 Introduction

Consolidation and aggregation of commodities is central to the effectiveness of supply chain networks. Whether it is the transportation for last-mile delivery or the movement of goods between warehouses, the consolidation of commodities can significantly reduce vehicle costs. However, this reduction in costs comes at the expense of increased complexity in the supply chain network by requiring commodities to be transported along multiple legs. The consolidation of commodities then relies on the effective coordination of long haul and local transportation vehicles at consolidation locations. It is the coordination of vehicles that makes the transportation of commodities in such a supply chain setting a challenging task—typically requiring the solution of numerous decision problems.

Commodities are regularly transported between warehouses to facilitate the distribution of goods to different parts of the supply chain network. The industry partner of this project typically performs this transportation using a point-to-point network—relying solely on long haul vehicles. However, the geographical distribution and potential clustering of warehouse locations provides opportunities to deploy a mixture of local (intra-cluster) and long haul (inter-cluster) transportation. The clustering of warehouse locations and consolidation of commodities is expected to lead to a reduction in vehicle routing costs through the effective coordination of intra- and inter-cluster transportation.

This paper aims to develop mathematical programming techniques to find high quality solutions to an integrated location clustering and vehicle routing supply chain management problem. In achieving this aim, the contributions of this paper are as follows: We i) present the first mathematical formulation combining warehouse clustering with intra- and inter-cluster transportation routing problems. Given the complexity of the considered supply chain management problem, the mathematical formulation relaxes the intra-cluster routing to only consider vehicle usage. Building on the work of Hosoda et al. (2022), we ii) combine a variant of the previously developed iterative algorithm with a branch-and-bound algorithm used to solve the proposed relaxation. The Ubiquity Generator (UG) framework (UG version 1.0) is deployed to iii) develop a parallel heuristic that finds high quality solutions for the integrated supply chain

management problem. The developed solution approach will iv) demonstrate how exact and heuristic methods can be combined within effective parallel solution algorithms. Finally, v) the proposed algorithm will be shown to provide high quality solutions with provable bounds on optimality. We present a parallel algorithmic framework that is flexible and capable of being deployed to find high quality solutions to challenging large-scale mathematical programming problems.

This paper is structured as follows: Related work will be discussed in Section 2. Section 3 presents and defines the considered supply chain management problem integrating warehouse clustering with intra- and inter-cluster vehicle routing, named the supply chain service network design problem (SCSNDP). A detailed description of the SCSNDP and modelling approach is presented in Section 3.1. A major contribution of this paper is the mathematical formulation of the integrated supply chain management problem. This mathematical formulation—which is a relaxation of the SCSNDP—is presented in Section 3.2. The algorithmic contribution of this paper is the development of a parallel heuristic algorithm to find solutions for the SCSNDP. The proposed algorithm, termed the SCSNDP Relaxation Induced Search (SRIS), is presented in Section 4. Section 5 presents the computational experiments evaluating the performance of the parallel branch-and-bound algorithm. Finally, our conclusions are provided in Section 6.

2 Literature review

A major challenge in the formulation of integrated supply chain management problems is the coordination of numerous location and transportation decisions. In the context of the SCSNDP, such integration incorporates decisions that arise in location-routing (Drexl and Schneider, 2015; Nagy and Salhi, 2007), vehicle routing with synchronisation (Drexl, 2012), service network design (SNDP) (Crainic, 2000; Wieberneit, 2008), multi-echelon vehicle routing (Perboli et al., 2011) and vehicle routing with pickup and delivery (PDP) (Desaulniers et al., 2002; Savelsbergh and Sol, 1995). A particularly challenging feature of integrated supply chain management problems is that the individual location and routing problems can be difficult to model and solve in isolation. Thus, any integration involving the combination of location and routing decisions typically leads to large mathematical programming formulations, which can render exact solution techniques intractable (Drexl and Schneider, 2015; Nagy and Salhi, 2007). Further, the introduction of synchronisation between different vehicle types significantly increases the

complexity of the resulting mathematical programming formulations (Drexl, 2012).

2.1 Integrating clustering, transshipment and routing

Clustering, location and routing decisions at the core of the SCSNDP are fundamental to the location-routing problem (Drexl and Schneider, 2015; Nagy and Salhi, 2007). As highlighted in Nagy and Salhi (2007) and Drexl and Schneider (2015), many different modelling approaches have been proposed for location-routing problems—including mathematical programming formulations. Given the complexity of the resulting integrated problem, it is common to abandon mathematical programming formulations and develop logic-based heuristics. Such a heuristic based on a clustering approach for a multi-depot location-routing problem is discussed by Lam and Mittenthal (2013). Similarly, Barreto et al. (2007) discusses different clustering techniques in the formulation of an integrated facility location and vehicle routing problem. The modelling concepts presented by Barreto et al. (2007) and Lam and Mittenthal (2013) motivate the integration of clustering with the intra- and inter-cluster transportation problems.

An important feature of the SCSNDP, commonly omitted from location routing problems, is the transportation of commodities between clusters. Addressing this issue, Contardo et al. (2012) present a mathematical programming formulation integrating the multi-echelon vehicle routing and location-routing problem. In a more complex setting, Wang et al. (2018) investigates the clustering of customers for last-mile delivery in a multi-echelon location-routing problem. While Drexl and Schneider (2015) discuss the multi-echelon problem with general routing decisions between levels, typical problem descriptions involve only the point-to-point movement of vehicles between levels. As such, there is little investigation into problems that involve the transfer of commodities between warehouses located on the same level, such as that which occurs in the SCSNDP.

The adaptiveness of transportation networks induced by clustering is embraced in the novel vehicle routing problem investigated by Salama and Srinivas (2020). Specifically, integrating truck and drone routing the truck stopping locations represent the consolidation locations and the drone then completes the last-mile delivery to the customer locations assigned to that cluster. This application has elements of intra- and inter-cluster routing; however, the latter, which is completed by a drone, only involves point-to-point transportation to and from the supporting truck.

An topic of research that is particularly relevant for the SCSNDP is the integration of

facility location and network design. An early example of a mathematical formulation for the integrated facility location and network design problem is presented by Melkote and Daskin (2001). An extensive investigation by Contreras and Fernández (2012) highlights the many variants and corresponding mathematical formulations for problems combining facility location and network design. Many of the concepts discussed by Contreras and Fernández (2012) are relevant for the SCSNDP. However, by incorporating routing decisions in the SCSNDP, the problem complexity is much greater than the problems discussed by Contreras and Fernández (2012).

Given a set of consolidation locations, the transportation schedule to effectively move commodities between these locations is found by solving an SNDP. Such a problem is a subproblem in the integration of location and routing proposed in the SCSNDP. Extensive reviews of the SNDP have been presented by Crainic (2000) and Wieberneit (2008). Given that the SNDP purely focuses on the transportation of commodities between consolidation locations, there has been increasing interest in the integration with first- and last-mile transportation. The most prominent examples integrating long-haul with first- and last-mile transportation are presented by Medina et al. (2019), Wolfinger et al. (2019) and Heggen et al. (2019). The most common feature of previous work is the synchronisation of vehicles when transferring commodities between long-haul and local transportation. Typically, the local transportation regions are fixed; however, an exception to this is Heggen et al. (2019) where the warehouse clusters are determined based on demand.

A rich variant of the vehicle routing problem (VRP), which has also inspired many subvariants, is the PDP. For a general overview of the PDP, the reader is referred to Savelsbergh and Sol (1995) and Desaulniers et al. (2002). The PDP is encountered as a subproblem of the SCSNDP, which is solved to identify intra-cluster routes. Variants of the PDP that share the most similarities with the SCSNDP are those that incorporate cross-docking opportunities, as presented by Wen et al. (2009), Santos et al. (2013) and Petersen and Røpke (2011). However, the modelling of the PDP in these previous works only considers a single cross-docking opportunity. A more general formulation of the PDP with cross-docking opportunities is presented by Buijs et al. (2014). Many challenges associated with the synchronisation of vehicles are highlighted by Buijs et al. (2014). Such issues are addressed in the development of the mathematical formulation and solution approaches proposed for the SCSNDP.

Another characteristic of the SCSNDP that is discussed in previous work is the transship-

ment of commodities. In the context of the PDP, transshipment is considered in the mathematical models developed by Mitrović-Minić and Laporte (2006) and Rais et al. (2014). More recently, Wolfinger (2021) presents a comprehensive mathematical model for the PDP with transshipment that incorporates time windows and split loads.

An in-depth survey on the use of synchronisation constraints in vehicle routing problems is presented by Drexl (2012). While many different types of synchronisation are discussed, the most relevant for the SCSNDP is operational synchronisation. The handling of operational synchronisation between intra- and inter-cluster transportation routes is a focus of the mathematical model developed for the SCSNDP.

2.2 Solution algorithms

Various solutions methods are proposed to solve the supply chain management problems discussed above. While exact solutions methods are considered in many cases, typically the complexity of the mathematical models means that such methods can only be deployed for small instances. General purpose solvers are deployed successfully by Melkote and Daskin (2001), Perboli et al. (2011), Medina et al. (2019), Rais et al. (2014) and Salama and Srinivas (2020). A branch-and-price approach is developed by Santos et al. (2013) to solve the PDP with cross-docking. In an attempt to address the size of the SNDP, Medina et al. (2019) propose an exact solution method based on Dynamic Discretisation Discovery, first proposed by Boland et al. (2017).

Heuristic methods are a popular approach for solving supply chain management problems. In addition to using a general purpose solver, Salama and Srinivas (2020) develop a heuristic method to solve the integrated vehicle and drone delivery problem. The location-routing problems considered by Barreto et al. (2007) and Lam and Mittenthal (2013) are solved using purpose built heuristics. A meta-heuristic based on a genetic algorithm is developed by Wang et al. (2018) the multi-echelon routing problem. Wen et al. (2009) propose a tabu search algorithm to solve a variant of the PDP. A computationally effective large-neighbourhood search heuristic for the PDP was developed by Petersen and Røpke (2011). The challenge of solving an integrating long-haul and local transportation problem is addressed by Heggen et al. (2019) and Wolfinger et al. (2019) with the development large neighbourhood search and iterated local search algorithms, respectively.

Variants of the branch-and-bound framework have been developed to solve challenging sup-

ply chain management problems. A novel algorithm combining branch-and-cut and large neighbourhood search is proposed by Contardo et al. (2012) to solve an integrated multi-echelon vehicle routing and location-routing problem. A heuristic branch-and-bound algorithm for solving the travelling salesman problem with a drone is developed by Poikonen et al. (2019). In the context of capacitated network design, Holmberg and Yuan (2000) present a heuristic branch-and-bound approach that employs a Lagrangian heuristic to find upper and lower bounding solutions. This combination of exact and heuristic methods is a motivation for the methods developed in this paper to solve the SCSNDP.

3 The supply chain service network design problem

The SCSNDP considers the problem of determining low cost transportation routes that satisfy all pickup and delivery requests between a set of warehouses. Key characteristics of the problem setting is that there is no centralised consolidation location and the warehouses are located throughout a large geographical area. However, all warehouses are configurable as consolidation locations. As such, commodities may be transported on one of more vehicle routes with transshipment occurring at warehouses that are selected as consolidation locations.

There are a number of different decisions that must be made to solve the SCSNDP. First, the set of consolidation locations is not known a priori and are selected to best satisfy the pickup and delivery requests. Each warehouse is either designated as a consolidation location or within a cluster surrounding a consolidation location. Second, vehicle routes must be determined to transfer each commodity either i) between their pickup and delivery locations within a cluster or ii) from the pickup origin to a consolidation location, between consolidation locations and then from a consolidation location to the delivery destination. These routes are designated as either intra- or inter-cluster routes, where the former correspond to pickup and delivery routes and the latter are likened to trunk routes. The problem considered in this paper aims to cluster warehouses and select consolidation locations while determining intra- and inter-cluster routes to satisfy all pickup and delivery requests.

3.1 Modelling of the SCSNDP

The naïve approach, and previous approach of the industry partner, for satisfying all pickup and delivery requests between warehouses was to use point-to-point transportation. This is the

equivalent of modelling the transportation network as a complete graph, $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, between all warehouse locations, such as that presented in Figure 1. The direct delivery of commodities in a point-to-point network is very resource intensive—resulting in high transportation costs. Further, such a network ignores the possibility of consolidating pickup and/or delivery requests associated with warehouse locations that are *close* (by some measure of distance) to each other.

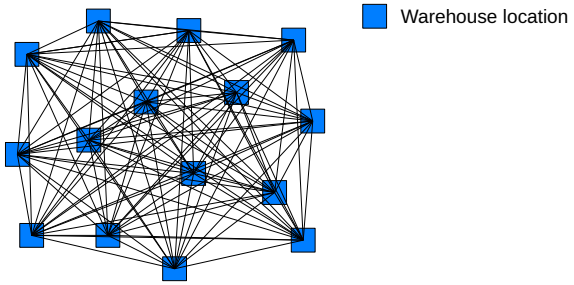


Figure 1: Point-to-point transportation network

An alternative approach for satisfying pickup and delivery requests, investigated by Hosoda et al. (2022), exploits the *closeness* of warehouses and the flexibility in designating consolidation locations. Central to this alternative approach is the clustering of warehouses, the design of an inter-cluster transportation network and determining intra-cluster pickup and delivery routes. Incorporating these three aspect results in a mathematical model for the SCSNDP that is formulated using a clustered transportation network, as shown in Figure 2.

Given the complexity of the SCSNDP, Hosoda et al. (2022) developed a heuristic approach that involved three mathematical models: the warehouse clustering problem, service network design problem and the pickup and delivery problem. In the following, the underlying ideas of these three mathematical models will be explained. For conciseness, we refrain from presenting the mathematical formulations and direct the reader to Hosoda et al. (2022) for a more in-depth

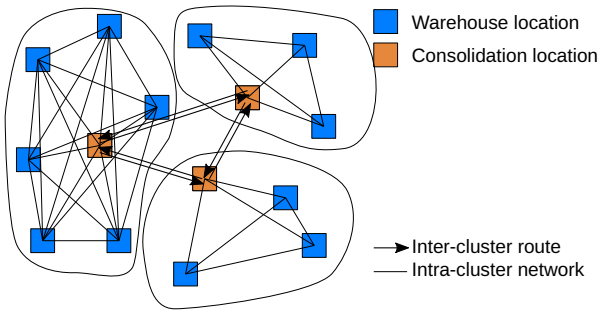


Figure 2: Clustered transportation network

discussion.

3.1.1 Warehouse clustering

The warehouse clustering problem (WCP) of Hosoda et al. (2022) is solved to form the intra- and inter-cluster transportation networks. Given a set of warehouse locations the WCP identifies $\bar{\gamma}$ consolidation locations from all warehouse locations and assigns all other warehouses to exactly one consolidation location. The consolidation location selection and warehouse assignment minimises some distance function—finding warehouse locations that are *close* to a consolidation location. In Hosoda et al. (2022) and this paper, the Haversine distance is used as the measure of *closeness*. An example solution to the WCP is presented in Figure 3.

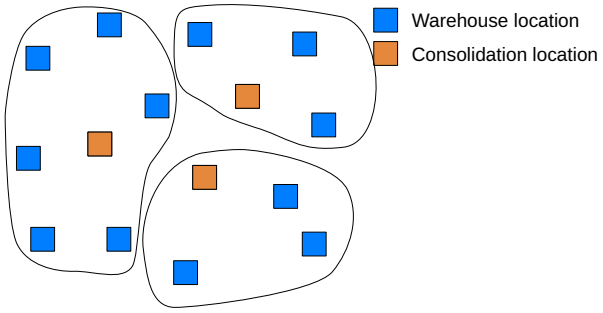


Figure 3: Warehouse clustering

3.1.2 Inter-cluster transportation

The WCP identifies a set of consolidation locations, between which inter-cluster routes are established. An example of the inter-cluster transportation network is presented in Figure 4. The inter-cluster transportation network is used to transport commodities that have pickup and delivery locations in different clusters. Such commodities are termed *out of cluster commodities* (OCC).

A service network design problem (SNDP) is solved to determine a transportation schedule for the inter-cluster routes. Consider a transportation network $\bar{\mathcal{G}} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$ and a set of commodities $\bar{\mathcal{K}}$ that must be transported between consolidation locations $\bar{\mathcal{N}}$ using the inter-cluster routes $\bar{\mathcal{A}}$. The solution to the SNDP identifies a vehicle schedule and the capacity of the arcs that is required to transport all commodities from their origin cluster to their destination cluster. The capacity of the arcs is relative to the number of vehicles traversing the arc at the same time. The vehicle schedule designates the departure and arrival times of all OCCs

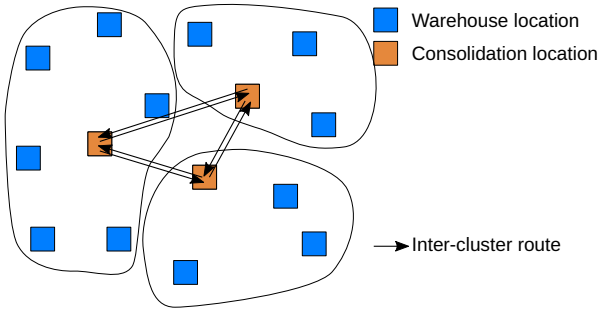


Figure 4: Inter-cluster network

at consolidation locations. These times must synchronise with intra-cluster routes to feasibly satisfy the OCC pickup and delivery requests. The SNDP is solved to minimise the number of vehicles, using a fixed cost for each vehicle used, and the cost of transporting commodities between consolidation locations.

3.1.3 Intra-cluster transportation

The intra-cluster networks are constructed such that the set of warehouses within each cluster are completely connected, but warehouses from different clusters are disconnected (excluding the inter-cluster routes). The intra-cluster networks for three clusters are presented in Figure 5. These networks are used to transport commodities either i) between locations within the cluster, ii) from a pickup origin to the consolidation location or iii) from the consolidation location to the delivery destination. Most importantly, the transportation of commodities to and from a consolidation location on intra-cluster routes must synchronise with the inter-cluster transportation schedule.

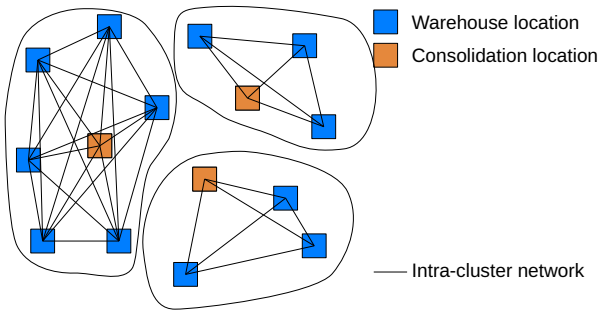


Figure 5: Intra-cluster network

The intra-cluster transportation routes are found by solving a vehicle routing problem with pickup and delivery. Let $\mathcal{G}^r = (\mathcal{N}^r, \mathcal{A}^r)$ denote the intra-cluster network for cluster r and the

set of vehicles available in cluster r are contained in the set $\mathcal{V}^r$. The consolidation location is both an origin and destination location, and is denoted as $\hat{o}^r$ and $\hat{d}^r$ respectively. All vehicles originate and terminate at the consolidation location. Let $\mathcal{K}^r$ denote the set of commodities that have a pickup or delivery location, $\hat{o}_k^r$ or $\hat{d}_k^r$, within cluster r. If the pickup location is within a cluster other than r, then $\hat{o}_k^r$ set to $\hat{o}^r$. Similarly, if the delivery location is within a cluster other than r, then $\hat{d}_k^r$ is set to $\hat{d}^r$. The PDP is then solved to find a set of vehicle routes that originate from $\hat{o}^r$ and terminate at $\hat{d}^r$ that collectively satisfy all pickup and delivery request within cluster r. The identified vehicle routes must also respect time windows, vehicle capacities and warehouse business hours.

3.2 Formulation of integrated problem

There are two features of the SCSNDP that introduce complications to the formulation of the fully integrated problem. The first is the formulation of the SNDP without prior knowledge of the consolidation locations. As a result, the underlying graph for the time-space network used in the SNDP is constructed as a complete graph between all warehouse location for every time step. The second complication is the ability to solve the PDP for each individual cluster. Since the PDP is a difficult problem to solve in isolation, its inclusion in the integrated problem significantly increases the complexity of the SCSNDP. In response to these challenges, a complete formulation of the SCSNDP is not proposed in this paper, but a relaxation is presented that will provide valid lower bounds.

The relaxation of the SCSNDP is formulated using a complete graph connecting all warehouse locations. The decisions of the WCP, SNDP, and PDP select the edges that will be used for intra- and inter-cluster transportation. The underlying graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ is defined with the set $\mathcal{N}$ containing all warehouse locations i and $\mathcal{A}$ containing all edges (i, j) between those locations. Central to the SCSNDP is the transportation of commodities k between warehouses. The set of commodities is denoted by $\mathcal{K}$. Each commodity k has one pickup and one delivery location, denoted by o_k and d_k respectively.

3.2.1 Warehouse clustering problem

The WCP has two major decisions: identify the consolidation locations and determine the warehouses that are connected to each consolidation location. The variables z_j equal 1 to indicate that warehouse j is selected as a consolidation location, and 0 otherwise. While in

the algorithm proposed by Hosoda et al. (2022) the total number of clusters is fixed, in the formulation of the SCSNDP relaxation this is defined by the variable γ. The assignment of each warehouse i to consolidation location j is indicated by the variables w_{ij}. Finally, the binary variables λ_{ki} equal 1, if and only if, the pickup and delivery location of commodity k, given by o_k and d_k, are assigned to the same consolidation location i. Such commodities only require intra-cluster transportation.

The constraints defining the WCP within the integrated formulation of the SCSNDP are given by

$$\sum_{j \in \mathcal{N}} z_j = \gamma, \tag{1a}$$

$$\sum_{j \in \mathcal{N}} w_{ij} = 1 \quad \forall i \in \mathcal{N}, \tag{1b}$$

$$\sum_{i \in \mathcal{N}} w_{ij} \leq |\mathcal{N}| z_j \quad \forall j \in \mathcal{N}, \tag{1c}$$

$$\lambda_{ki} \leq w_{o_k i} \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \tag{1d}$$

$$\lambda_{ki} \leq w_{d_k i} \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \tag{1e}$$

$$\lambda_{ki} \geq w_{o_k i} + w_{d_k i} - 1 \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \tag{1f}$$

$$\gamma \in [2, \bar{\gamma}], \tag{1g}$$

$$z_j \in \{0, 1\} \quad \forall j \in \mathcal{N}, \tag{1h}$$

$$w_{ij} \in \{0, 1\} \quad \forall (i, j) \in \mathcal{A}, \tag{1i}$$

$$\lambda_{ki} \in \{0, 1\} \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}. \tag{1j}$$

Constraint (1a) sets γ to the number of selected consolidation locations. The restriction that each warehouse must be assigned to exactly one consolidation location is imposed by constraints (1b). Constraints (1c) ensures that warehouse i is assigned to warehouse j only if j is a consolidation location. The constraint set (1d)–(1f) are included to indicate whether the pickup and delivery locations of commodity k are assigned to the same consolidation location.

3.2.2 Service network design problem

The SNDP is solved to determine the inter-cluster transportation schedule between consolidation locations. Different to the description of the inter-cluster transportation described in Section 3.1.2, the underlying transportation network is unknown a priori. Thus, the trans-

portation network used to model the SNDP is a time expanded network based on the complete graph $\mathcal{G}$.

The time expanded network for the SNDP, denoted by $\mathcal{G}_\mathcal{T}$, is defined within the planning horizon bounded by E, which is the earliest departure of any commodity, and L, which is the latest arrival of any commodity. The discrete time points used in the time expanded network are given by the set $\mathcal{T} = \{E + m\Delta | m \in \mathbb{Z}_{\geq 0}, E + m\Delta < L\}$, where $\Delta > 0$ is the discretisation interval. The node set is given by the Cartesian product of $\mathcal{N}$ and $\mathcal{T}$, i.e. $\mathcal{N}_\mathcal{T} = \mathcal{N} \times \mathcal{T}$. Given two locations $i, j \in \mathcal{N}$, the minimum travel time between these locations is given by tt_{ij}. To provide flexibility in the travel times between i and j, the time set $\mathcal{T}_{ijt} = \{\lceil t + tt_{ij} + m\Delta \rceil_\Delta \,|\, m \in \mathbb{Z}_{\geq 0}, m\Delta \leq B\}$ is defined, where $\lceil \cdot \rceil_\Delta$ rounds the time up to the nearest discretisation interval and B is the maximum additional travel time. Thus, the arc set is given by $\mathcal{A}_\mathcal{T} = \{((i,t),(j,\bar{t})) \,|\, (i,j) \in \mathcal{A}, t \in \mathcal{T}, \bar{t} \in \mathcal{T}_{ijt}\}$. Finally, the inter-cluster transportation vehicles are permitted to wait at any location within the network, which is modelled using the arc set $\mathcal{H}_\mathcal{T} = \{((i,t),(i,t+\Delta)) \,|\, (i,t) \in \mathcal{N}_\mathcal{T}, t + \Delta \leq L\}$. Thus, the time expanded network underlying the formulation of the SNDP is given by $\mathcal{G}_\mathcal{T} = (\mathcal{N}_\mathcal{T}, \mathcal{A}_\mathcal{T} \cup \mathcal{H}_\mathcal{T})$.

Since the clustering and consolidation locations are not known a priori, the departure and arrival consolidation locations and respective times must be determined for each commodity. As such, the variables h_{ti}^{ek} and h_{ti}^{lk} are defined to indicate whether commodity k must depart from, respectively arrive at, consolidation location i after, respectively before, time t. The variables h_{ti}^{ek} are set to 1 only if the origin of commodity k is assigned to the consolidation location i, and similarly for h_{ti}^{lk} with respect to the destination of k. Note that if the origin and destination of commodity k are within the same cluster, then k does not need to be transported on any inter-cluster routes.

The decision variables of the SNDP will identify the capacity of the inter-cluster routes in order to transport commodities between consolidation locations. The variables $x_{ij}^{kt\bar{t}}$ equal 1 if commodity k is transported between consolidation locations i and j, departing at t and arriving at $\bar{t}$. The movement of an inter-cluster vehicle between i and j departing at t and arriving at $\bar{t}$ is indicated by the variables $y_{ij}^{t\bar{t}}$ being set to 1. Finally, the variables θ_k are introduced to indicate that commodity k is transported directly from origin to destination using third-party vehicles, which is termed direct delivery.

The constraints of the SNDP that are included within the integrated formulation of the

SCSNDP are given by

$$w_{o_k i} - \lambda_{ki} \leq \sum_{t \in T} h_{ti}^{ek} + \theta_k \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{2a}$$

$$w_{o_k i} - \lambda_{ki} \geq \sum_{t \in T} h_{ti}^{ek} \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{2b}$$

$$w_{d_k i} - \lambda_{ki} \leq \sum_{t \in T} h_{ti}^{lk} + \theta_k \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{2c}$$

$$w_{d_k i} - \lambda_{ki} \geq \sum_{t \in T} h_{ti}^{lk} \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{2d}$$

$$\sum_{((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T} \cup \mathcal{H}_\mathcal{T}} x_{ij}^{kt\bar{t}} - \sum_{((j,\bar{t}),(i,t)) \in \mathcal{A}_\mathcal{T} \cup \mathcal{H}_\mathcal{T}} x_{ji}^{k\bar{t}t} = h_{ti}^{ek} - h_{ti}^{lk} \quad \forall k \in \mathcal{K}, (i,t) \in \mathcal{N}_\mathcal{T}, \tag{2e}$$

$$\sum_{k \in \mathcal{K}} q_k x_{ij}^{kt\bar{t}} \leq U y_{ij}^{t\bar{t}} \quad \forall ((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}, \tag{2f}$$

$$\sum_{k \in \mathcal{K}} \left\{ \sum_{((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T} \cup \mathcal{H}_\mathcal{T}} x_{ij}^{kt\bar{t}} + \sum_{((j,\bar{t}),(i,t)) \in \mathcal{A}_\mathcal{T} \cup \mathcal{H}_\mathcal{T}} x_{ji}^{k\bar{t}t} \right\}$$

$$\leq |\mathcal{K}||\mathcal{A}_\mathcal{T}| z_j \quad \forall j \in \mathcal{N}, \tag{2g}$$

$$x_{ij}^{kt\bar{t}} \in \{0,1\} \quad \forall ((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}, k \in \mathcal{K}, \tag{2h}$$

$$y_{ij}^{t\bar{t}} \in \mathbb{Z}_{\geq 0} \quad \forall ((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}, \tag{2i}$$

$$h_{ti}^{ek}, h_{ti}^{lk} \in \{0,1\} \quad \forall k \in \mathcal{K}, \forall (i,t) \in \mathcal{N}_\mathcal{T}, \tag{2j}$$

$$\theta_k \in \{0,1\} \quad \forall k \in \mathcal{K}. \tag{2k}$$

Constraints (2a) and (2b) ensure that if the origin and departure of commodity k are in different clusters, then it must either depart from consolidation location i or be transported using a third-party vehicle. Similarly, the arrival of commodity k at consolidation location i or its transportation using a third party vehicle is enforced by constraints (2c) and (2d). The flow balance of commodities through the network induced by the consolidation locations is given by constraints (2e). Constraints (2f) ensure that sufficient vehicle capacity is available to transport commodities between i and j. Finally, constraints (2g) ensure that only consolidation locations are used by inter-cluster transportation vehicles.

3.2.3 Pickup and delivery problem

The relaxation of the SCSNDP considered in this paper omits the routing component of the PDP. As such, the PDP of the SCSNDP is modelled as a packing problem with side constraints. The variables u_{ki}^v are defined to equal 1 if the pickup or delivery of commodity k is assigned

to vehicle v that departs from consolidation location i. The use of vehicle v departing from consolidation location i is indicated by the variables α_i^v. Each commodity k has an order quantity q_k and the capacity of the intra-transportation vehicle v is given by Q^v.

The linking between the SNDP and PDP is given through the departure and arrival of the intra-cluster vehicles at the consolidation locations. The departure and arrival times of intra-cluster vehicle v from consolidation location i are given by $\bar{T}_i^v$ and $\hat{T}_i^v$ respectively. Note that in the relaxation of the PDP proposed in this paper, the vehicle departure and arrival times at intermediate stops are not considered. However, the pickup and delivery time windows for commodity k, given by $[e_k^p, l_k^p]$ and $[e_k^d, l_k^d]$ respectively, must be respected when considering the vehicle departure and arrival times from the consolidation locations. While the intra-cluster vehicle routes are not explicitly identified, the total travel time of each vehicle is estimated by summing the shortest edge entering each assigned pickup or delivery location, which is denoted by the parameter $\hat{tt}_k$.

Considering the time windows and travel times between locations, it is possible that there exists pairs of commodities that can not be assigned to the same vehicle. This is due to a conflict between the pickup/delivery times and the travel time between the commodities origins/destinations. The set of commodities with time window/travel time conflicts is denoted by $\bar{\mathcal{K}}$.

The constraints that define the relaxation of the PDP are given by

$$w_{o_k i} \leq \sum_{v \in \mathcal{V}} u_{ki}^v + \theta_k \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N} \backslash \{o_k\}, \tag{3a}$$

$$w_{d_k i} \leq \sum_{v \in \mathcal{V}} u_{ki}^v + \theta_k \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N} \backslash \{d_k\}, \tag{3b}$$

$$\bar{T}_i^v \geq \sum_{t \in \mathcal{T}} th_{ti}^{lk} - M(1 - u_{ki}^v) - M \sum_{j \in \mathcal{N}} \lambda_{kj} - M(1 - w_{d_k i}) \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3c}$$

$$\bar{T}_i^v \leq l_k^p w_{o_k i} + l_k^d w_{d_k i} + M(1 - u_{ki}^v) \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3d}$$

$$\hat{T}_i^v \leq \sum_{t \in \mathcal{T}} th_{ti}^{ek} + M(1 - u_{ki}^v) + M \sum_{j \in \mathcal{N}} \lambda_{kj} + M(1 - w_{o_k i}) \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3e}$$

$$\hat{T}_i^v \geq e_k^p w_{o_k i} + e_k^d w_{d_k i} - M(1 - u_{ki}^v) \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3f}$$

$$\hat{T}_i^v \geq \bar{T}_i^v + \sum_{k \in \mathcal{K}} \hat{tt}_k u_{ki}^v \quad \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3g}$$

$$\sum_{k \in \mathcal{K}} q_k u_{ki}^v \leq Q^v \alpha_i^v \quad \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3h}$$

$$\sum_{k\in\mathcal{K}} \hat{tt}_k u^v_{ki} \leq T^v \alpha^v_i \quad \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3i}$$

$$\sum_{i\in\mathcal{N}} \alpha^v_i \leq 1 \quad \forall v \in \mathcal{V}, \tag{3j}$$

$$u^v_{ki} + u^v_{k'i} \leq 1 \quad \forall (k,k') \in \bar{\mathcal{K}}, \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3k}$$

$$u^v_{ki} \in \{0,1\} \quad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3l}$$

$$\alpha^v_i \in \{0,1\}, \quad \forall i \in \mathcal{N}, \forall v \in \mathcal{V}, \tag{3m}$$

$$\bar{T}^v_i, \hat{T}^v_i \geq 0 \quad \forall i \in \mathcal{N}, \forall v \in \mathcal{V}. \tag{3n}$$

Constraints (3a) and (3b) assign the pickup and delivery of commodity k to a vehicle v, except if commodity k is delivered using a third-party vehicle. If the pickup and delivery locations are in two different clusters, then the commodity will be transported on two intra-cluster vehicles, unless transported by direct delivery. The time window and travel time constraints are given by (3c)–(3g). Constraints (3c) ensures that vehicle v departs from consolidation location i after the arrival of commodity k on an inter-cluster vehicle. The departure of intra-cluster vehicle v must respect the latest pickup or delivery time of commodity k—depending on the cluster assignment of the commodity's origin and destination. These conditions are enforced by constraints (3d). Similar to the departure times, constraints (3e) and (3f) ensure vehicle v arrives at the consolidation location before the departure of commodity k on an inter-cluster vehicle and after the commodity's earliest pickup or delivery time—again, depending on the cluster assignment of the commodity's origin and destination. Constraints (3c)–(3f) are only active if commodity k is assigned to vehicle v departing from consolidation location i, i.e. $u^v_{ki} = 1$. Further, constraints (3d) and (3f) are only enforced if the origin and destination of commodity k are in different clusters, i.e. $\sum_{j\in\mathcal{N}} w'_{kj} = 0$. Finally, constraints (3d) are only relevant when the destination of commodity k is assigned to consolidation location i, and similarly for constraints (3f) with respect to the commodity's origin. The use of vehicle v is indicated by constraints (3h) and (3i), which also impose the vehicle capacity Q^v and maximum travel time T^v. Finally, the conflict of assigning commodities k and k' to vehicle v, if $(k,k') \in \bar{\mathcal{K}}$ is enforced by constraints (3k).

3.2.4 Objective function

The objective function is given by a linear combination of the objectives from the WCP, SNDP and PDP defined by Hosoda et al. (2022). The three components of the objective function for

the integrated SCSNDP formulation comprise:

- The cost of the WCP given by the distance between locations i and j, denoted by β_{ij}.
- The cost of the SNDP made up of a fixed cost f for each inter-cluster vehicle used and the sum of costs of transporting commodity k along arc (i, j), denoted by c_{ij}. The transportation cost is proportional to the order quantity q_k of commodity k.
- The cost of the PDP given by the sum of vehicle usage costs and travel time. The use of intra-cluster vehicle v incurs a fixed cost of f. The travel time costs for the PDP are relaxed to the weighted sum of the travel time of the shortest edge incident with the origin or destination of commodity k. The sum of the travel time is weighted by κ.

In addition to the above costs, the objective of the SCSNDP incorporates the costs associated with the direct delivery of commodities. This is a fixed cost denoted by ξ.

The objective function of the SCSNDP is given by

$$\begin{aligned}\sum_{(i,j)\in\mathcal{A}} \beta_{ij} w_{ij} + \sum_{((i,t),(j,\bar{t}))\in\mathcal{A}_\mathcal{T}} f y_{ij}^{t\bar{t}} + \sum_{k\in\mathcal{K}} \sum_{((i,t),(j,\bar{t}))\in\mathcal{A}_\mathcal{T}} c_{ij} q_k x_{ij}^{kt\bar{t}} \\ + \sum_{i\in\mathcal{N}} \sum_{v\in\mathcal{V}} f\alpha_i^v + \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}} \kappa \hat{tt}_k u_{ki}^v + \sum_{k\in\mathcal{K}} \xi\theta_k.\end{aligned} \tag{4}$$

4 Parallel algorithm combining exact and heuristic methods

The SCSNDP Relaxation Induced Search (SRIS) combines the relaxation, presented in Section 3.1, and the iterative heuristic developed by Hosoda et al. (2022) within a parallel framework. The relaxation is solved directly as a MIP to provide lower bounds on the optimal solution—to evaluate the quality of solutions found by the iterative heuristic. During the execution of the branch-and-bound algorithm, configurations of consolidation locations, extracted from fractional and integral solution to the relaxation, are used as input to the iterative heuristic. Extracting consolidation location configurations during the branch-and-bound algorithm is used as a method for diversifying the heuristic search for primal solutions. It is expected that such diversification will be superior to the random approach employed in the multi-armed bandit algorithm developed by Hosoda et al. (2022).

In the following sections, the core components of the proposed parallel algorithm for solving the SCSNDP will be described. First, the software architecture combining the relaxation and

iterative heuristic within a parallel algorithm will be described in Section 4.1. While the iterative heuristic is presented in Hosoda et al. (2022), an overview of the heuristic and details relevant for the parallel algorithm are discussed in Section 4.2. This will be followed by the descriptions of the sequential and parallel versions of the SRIS that combine the relaxation and iterative heuristic in Sections 4.3 and 4.4 respectively. These two different algorithms can be executed using the software architecture presented in Section 4.1.

4.1 Parallel solver software architecture

The parallel solver, labelled as ParaSCSNDP, has been designed to exploit concurrent solving of the relaxation and the execution of the iterative heuristic, while providing flexibility to balance the computational effort. This is achieved through the use of the Ubiquity Generator (UG) framework (UG version 1.0), which provides a set of base classes for *high-level task parallelisation* and a flexible parallelisation of branch-and-bound based solvers. For an overview of the UG framework Version 1.0 the reader is referred to Tateiwa et al. (2021). An overview of the parallel solver software architecture is presented in Figure 6.

The software architecture implementing ParaSCSNDP employs task-based parallelism. There are two types of tasks executed in ParaSCSNDP, which are labelled as the relaxation solver and the iterative heuristic. There are n solvers, which correspond to the available processors, to which these tasks can be assigned. The software architecture allows for any number (up to n) of relaxation solvers or iterative heuristic tasks to be created. However, at any one time there will be m active solvers, where $m \leq n$. For convenience, it is assumed that the relaxation

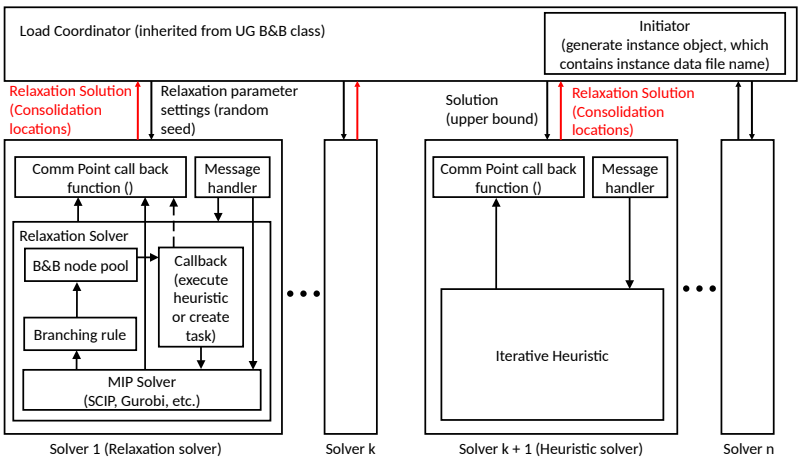


Figure 6: ParaSCSNDP architecture

solver is assigned to k solvers and $m - k$ solvers execute the iterative heuristic. Typically, the number of relaxation solvers k is fixed. The number of iterative heuristic tasks depends on the availability of consolidation location configurations identified by the relaxation solvers.

The design of ParaSCSNDP permits the transfer and distribution of both relaxation solver and iterative heuristic tasks. The distribution of tasks and the balancing of computational effort is performed in the LoadCoordinator. All tasks that are created are first transferred to the LoadCoordinator and placed in a queue. These tasks are then assigned to available solvers on demand. In the current design, the tasks distributed by the LoadCoordinator are iterative heuristic tasks—defined by consolidation location configurations.

A callback method for a general purpose solver is used to extract consolidation location configurations from integer and fractional solutions found during the execution of the relaxation solver. In Figure 6, the dashed line starting from the callback indicates two different modes of the relaxation solver. In the sequential mode, the consolidation location configuration is used to execute the iterative heuristic within the branch-and-bound algorithm. In the parallel mode, the consolidation location configuration is communicated to the LoadCoordinator as a task for distribution to another solver.

4.2 Iterative heuristic

An iterative heuristic is employed to find primal feasible solutions for the SCSNDP. An overview of the iterative heuristic is presented in Figure 7. The iterative heuristic builds upon a sequential heuristic for the SCSNDP. Briefly, the sequential heuristic first solves the WCP to determine a clustering of warehouses, then solves the SNDP to find inter-cluster transportation schedules and finally the PDP is solved to identify a set of intra-cluster routes. Since such a sequential heuristic is inherently suboptimal, and potentially leads to infeasible solutions, an iterative process between the SNDP and PDP is used to provide feedback between the intra- and inter-cluster transportation problems.

The iterative heuristic employed in this paper is a subprocess of the multi-armed bandit algorithm developed by Hosoda et al. (2022). For a more detailed discussion of the algorithm—including the mathematical formulations for the WCP, SNDP, and PDP and respective solution methods for the mathematical programs—the reader is referred to Hosoda et al. (2022).

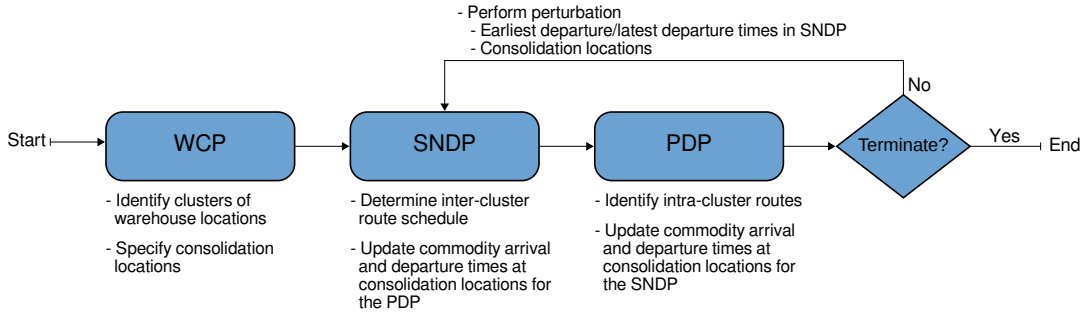


Figure 7: Iterative heuristic

4.2.1 Feedback between the SNDP and PDP

The most important part of the iterative heuristic is the feedback between the SNDP and PDP. This feedback is given through the commodity departure and arrival times at the consolidation locations and the synchronisation of intra- and inter-cluster transportation. Each OCC must arrive at their origin consolidation location on an intra-cluster route prior to the departure of that commodity on an inter-cluster vehicle. Similarly, intra-cluster routes that deliver an OCC must depart after the arrival of that commodity on an inter-cluster vehicle. Any violation of the departure and arrival times of OCCs, with respect to the intra- and inter-cluster transportation routes, is penalised. By iterating between the SNDP and PDP, the penalties—and corresponding violations—are reduced to zero. The connection between the SNDP and PDP in the iterative algorithm and the modifications required to impose the time-window violation penalties are detailed in Hosoda et al. (2022).

Penalising departure and arrival violations in the SNDP Within the SNDP, each OCC k is assigned an earliest departure $\bar{e}_k$ and latest arrival $\bar{l}_k$ times at the origin $\bar{o}_k$ and destination $\bar{d}_k$ consolidation locations, respectively. The times $\bar{e}_k$ and $\bar{l}_k$ are given by the arrival and departure of intra-cluster routes for the first- and last-mile transportation, respectively, of commodity k. A penalty is imposed for the departure of k from $\bar{o}_k$ on an inter-cluster vehicle prior to $\bar{e}_k$. Similarly, the arrival of k on an inter-cluster vehicle at $\bar{d}_k$ after $\bar{l}_k$ incurs a penalty. The early departure penalty is defined as $\phi_t^k = Y(\bar{e}_k - t)^2$ and the late arrival penalty is defined as $\theta_t^k = Y(t - \bar{l}_k)^2$, where t is the actual departure/arrival time and Y is a constant penalty factor.

Penalising time window violations in the PDP There are two types of time windows that are considered in the PDP. The first are specified in the instance data for all warehouse locations, which dictate the pickup and delivery times of commodity k. The second are given by the solution to the SNDP—corresponding to the arrival and departure of each OCC on inter-cluster vehicles. In the solution algorithm for the PDP, both types of time windows are transformed into soft constraints to permit flexibility in the arrival and departure of intra-cluster vehicles. Any violation in the soft constraints is penalised in the objective of the PDP. Given that the first type of time windows are defined by the input data, their satisfaction takes a higher priority in the solution algorithm. As such, each time unit violation of the first type of time window incurs a penalty of $10Y$, while a unit violation of the second type incurs a penalty of $9Y$.

4.2.2 Updating pickup and delivery time windows

The solution to the SNDP identifies an inter-cluster transportation schedule that is used to define the time windows for the departure and arrival of intra-cluster routes. Consider commodity k and its transportation on an inter-cluster route between consolidation locations i and j. If k departs i at time t, then when solving the PDP the arrival time window for k at i is set to $[E, t]$. Similarly, if k arrives at j at time t', then the departure time window for the intra-cluster route transporting k from j is set to $[t', L]$.

Since the intra-cluster transportation networks are separated into the clustered regions, the solution to the PDP is used to modify the time windows for the arrival and departure of OCCs from each region. Consider commodities k and k', which have their respective origin and destination in region r. Let i denote the consolidation location of region r. If k is transported on an intra-cluster vehicle from its origin to i, arriving at t, then the earliest departure time for k in the SNDP is set to $\lfloor t \rfloor_\Delta$. Similarly, if k' departs i at t' on an intra-cluster vehicle travelling to its destination, then the latest arrival time for k'' in the SNDP is set to $\lceil t' \rceil_\Delta$. The functions $\lceil \cdot \rceil_\Delta$ and $\lfloor \cdot \rfloor_\Delta$ round the time up and down, respectively, to the nearest discretisation interval based on Δ. Note that the earliest departure and latest arrival times in the SNDP are updated in each iteration of the algorithm regardless of whether there is a time window violation.

4.2.3 Perturbation techniques

The iterative algorithm quickly converges to a local optimal solution for the SCSNDP. As such, to search a larger neighbourhood and potential find improving feasible solutions, perturbation techniques are necessary to escape regions of local optima.

Shifting the earliest departure and latest arrival times The time windows for the SNDP are perturbed when it is identified that the iterative heuristic has reached a local optimal solution. This is indicated by the stalling of the algorithm, which is defined as:

Definition 1. *For iteration q, let Z_q denote the objective value of the SCSNDP. Given a memory of Q iterations, let $\mathcal{Q} := \{Z_{q-1}, Z_{q-2}, \ldots, Z_{q-Q}\}$ be the set of objective values for the SCSNDP stored in memory. The algorithm is identified as having stalled in iteration q if*

$$Z_q \in \mathcal{Q} \quad \textit{or} \quad Z_q > \max \mathcal{Q}.$$

When the iterative heuristic has stalled, the departure and arrival times of the OCCs at the consolidation locations—given by the solution to the PDP—are perturbed prior to the next execution of the SNDP (see Figure 7). Consider the set of departure and arrival times for the OCCs in each region r, denoted by $\mathcal{T}_D^r$ and $\mathcal{T}_A^r$. From each of $\mathcal{T}_D^r$ and $\mathcal{T}_A^r$, $\rho\%$ of the times are selected as candidates for perturbation. Each selected time is then shifted by t', where t' is sampled from a normal distribution with mean 0 and standard deviation Δ.

Changing the consolidation locations A larger change to the problem setting for the iterative algorithm is induced by changing consolidation locations within the given clusters. This larger change is triggered if there has been no improvement in the objective value for 2 iterations. Changing the consolidation locations is the performed as follows: First, a cluster r is randomly selected with the probability $|\mathcal{N}^r|/|\mathcal{N}|, r \in \mathcal{R}$, where $\mathcal{N}^r$ are the locations assigned to cluster r. Then a consolidation location is selected uniformly at random from the set $\mathcal{N}^r$. The iterative solving of the SNDP and PDP continues with the updated consolidation locations.

Following the perturbation of the consolidation locations, if an improvement in the objective value is not achieved after 2 iterations then the consolidation locations corresponding to the best solution are restored. One subsequent iteration is performed with the *best* consolidation locations, and then the perturbation procedure is executed again.

4.2.4 Termination conditions

The iterative heuristic terminates if either: i) a specified time limit is reached; ii) the *best* consolidation locations are restored after the consolidation location perturbation procedure has been performed twice; or iii) if the best objective value has been encountered 20 times. When the algorithm terminates, the objective value of the best solution is returned to the branch-and-bound algorithm.

4.3 Sequential SRIS

The balancing of computational effort between solving the relaxation and executing the iterative heuristic is critical for the performance of the sequential SRIS. As such, two different formulations of the relaxation, with increasing complexity, are solved. Additionally, the iterative heuristic is only deployed at select points during the relaxation solving process.

4.3.1 Relaxation formulations

One particular challenge that comes from solving the SCSNDP is the large number of constraints that are a result of unknown warehouse clustering and consolidation locations. This is particularly evident in the formulation of the SNDP, given by constraints (2). A two-stage approach for solving the relaxation is designed to address this issue. Initially, the relaxation is formed with constraints (1) and (3)—termed the WCP-PDP relaxation. In the WCP-PDP relaxation, constraints (3c)–(3g) are also relaxed, since they depend on values from the SNDP.

If the WCP-PDP relaxation is solved to optimality, then constraints (3c)–(3g) and the SNDP constraints are reintroduced. The full SCSNDP relaxation is solved to find a tighter bound on the objective for the SCSNDP.

4.3.2 Executing the iterative heuristic

The iterative heuristic is potentially executed when either i) the LP relaxation is solved at a node, or ii) a new feasible solution is found for the SCSNDP relaxation. These two points are identified using callback functions available within general purpose MIP solvers. When either of these executing points are reached, a configuration for the consolidation locations is extracted from the LP or integral solution respectively. In the case of fractional LP solutions, the values of the $z_j, \forall j \in \mathcal{N}$ and γ are rounded to the nearest integer. The set of consolidation locations

is then given by $\bar{\mathcal{N}} = \{j \,|\, \lceil z_j \rfloor = 1\}$. If $|\bar{\mathcal{N}}| > \lceil \gamma \rfloor$, then elements of $\bar{\mathcal{N}}$ are randomly removed such that $|\bar{\mathcal{N}}| = \lceil \gamma \rfloor$. In the case of an integer feasible solution, then $\bar{\mathcal{N}} = \{j \,|\, z_j = 1\}$.

Since the input to the iterative heuristic depends on a subset of the decision variables for the SCSNDP relaxation, it is possible that the same configuration of consolidation locations is encountered more than once. In such cases, the consolidation locations are perturbed to form a different configuration. Specifically, $j' \in \bar{\mathcal{N}}$ is randomly selected and is replaced by j'' that is randomly selected from $\mathcal{N} \backslash \bar{\mathcal{N}}$. This new configuration is then checked against previous configurations. If the new configuration is not unique then the perturbation is performed again. This perturbation process is executed at most $2|\mathcal{N}|$ times for each configuration. If a unique configuration is not found, then the iterative heuristic is not executed for the given solution.

The frequency at which the iterative heuristic is executed, as defined by the number of LP and integer solutions found, is dynamically adjusted during the branch-and-bound search. Specifically, the iterative heuristic is executed only after η LP and integer solutions are encountered. Initially $\eta = 1$, meaning that the next solution found is used as input to the iterative heuristic. If the heuristic fails to improve the objective value for the SCSNDP, then η is increased by a factor of 2, i.e. $\eta = 2\eta$. If the heuristic improves the objective value for the SCSNDP, then η is reset to 1. This dynamic frequency helps balance the computational effort devoted to solving the relaxation and the execution of the iterative heuristic.

4.4 Parallel SRIS

The parallel SRIS aims to use multi-processor computational resources to accelerate the search for both upper and lower bounds for the SCSNDP. Designed to achieve this goal, the core feature of the algorithm is the concurrent solving the relaxation and iterative algorithm. To balance the computational effort between the relaxation and iterative algorithm, the parallel SRIS executes two different phases. The first is a racing phase (see Shinano et al. (2018) for details) that identifies the best settings for the relaxation solver. The second devotes more effort to the iterative heuristic to accelerate the search for primal feasible solutions.

The racing phase deploys relaxation solvers to all available processors—demonstrated in Figure 8. In this phase, all relaxation solvers are executed in sequential mode (described in Section 4.3). Thus, the callback method executes the iterative heuristic within the branch-and-bound search of the SCSNDP relaxation.

There is very little communication between the solvers and LoadCoordinator during

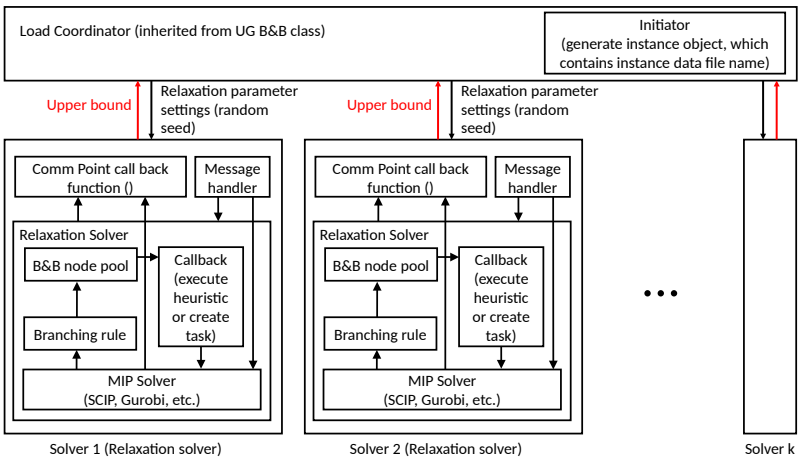


Figure 8: ParaSCSNDP– racing phase

the racing phase. Specifically, the initial setup for the relaxation solvers requires parameter settings that are communicated from the LoadCoordinator. In addition, throughout the solving of the relaxation, any improvement in the upper bound is communicated back to the LoadCoordinator. Other than these two points of communication, the relaxation solvers are executed in isolation.

A unique set of parameters is provided to each of the relaxation solvers. This aims to diversify the search in the branch-and-bound algorithm, leading to a broad collection of configuration locations to be supplied to the iterative heuristics. While any parameter settings can be provided, in the current implementation only the random seed provided to the MIP solver differs between the relaxation solvers. Different random seeds have been observed to provide sufficient variation in the solving process for the SCSNDP relaxation.

The racing phase terminates when the iterative heuristic has been executed a prespecified number of times by all solvers. In the current implementation this limit is set to 5. At the end of the racing phase, the 3 solvers with the best found primal bounds are selected as winners. The winning solvers continue solving the relaxation, while the remaining solvers are terminated. This triggers the start of the search phase, where the available processors are then used for the concurrent execution of the iterative heuristic.

The search phase solves the relaxation and the iterative heuristic concurrently. The three relaxation solvers switch to the parallel execution mode. This involves a change in the callback method where the consolidation location configurations are now passed to the LoadCoordinator for input to the iterative heuristic on another solver. During the search phase, the

dynamic frequency mechanism described in Section 4.3.2 is not used. Instead, for each consolidation location configurations found, the difference from all previously found configurations is computed. If a consolidation location configuration has a hamming distance of at least 2 from all previously found configurations, then it is passed to the LoadCoordinator for input to the iterative heuristic.

In the search phase, the consolidation locations configuration communicated to the LoadCoordinator are stored in a queue. Along with the consolidation locations configuration, the lower bound for the relaxation is also communicated to the LoadCoordinator. This bound is used to prioritise (lower bound preferred) the selection of configurations for input to one of the iterative heuristic solvers.

During the iterative heuristic is executed, the solution and objective values are not communicated back to any relaxation solver. Only the objective values of the best found solutions are communicated to the LoadCoordinator from the iterative heuristic solvers. The communicated objective values are used to report the progress of the computation.

5 Computational experiments

The computational experiments evaluate the effectiveness of the SRIS at finding high quality solutions for the SCSNDP. Initially, the sequential SRIS is compared against the multi-armed bandit algorithm (MAB) developed by Hosoda et al. (2022). This will highlight the benefit of guiding the search for primal feasible solutions using the branching decisions from the relaxation. The second set of experiments evaluate the performance of the parallel SRIS. The run time of the algorithm and the optimality gaps of the found solutions are evaluated. Finally, the parallel performance—with respect to the processor idle time—is assessed.

The solution algorithm proposed in the paper has been implemented in C++. The parallel framework is implemented using UG version 1.0. The relaxation from the sequential and parallel SRIS and all MIPs in the iterative algorithm are solved using Gurobi 9.0.1 (Gurobi). The computational experiments have been performed on a computational cluster comprised of Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz CPUs and 125GB RAM per node.

5.1 Problem instances

A set of problem instances have been generated to be representative of the business practices of the industry partner. These are smaller than the instances sizes encountered in practice, but have been used as a proof of concept for the proposed solution algorithm. The smallest instances are generated to aid the evaluation of the parallel SRIS and assess the quality of solutions found.

The instance data is generated with a time horizon of 2 days. This matches the requirements of the industry partner were all pickup and delivery requests are completed within a 2 day period. All of the warehouses and vehicles must respect the business hours, which are from 6:00 until 20:00.

The warehouse locations are randomly distributed within a square or subregions within a square. The latter distribution mimics the natural clustering of locations within highly populated areas. Instances are created with $N \in \{5, 10, 25\}$ locations with the number of subregions C given so that they are realistic relative to the number of location. As such, for $N = 5$ and 10, $C \in \{1, 2\}$; and for $N = 25$, $C \in \{1, 5\}$ subregions.

The travel time between locations i and j, denoted by tt_{ij}, is selected uniformly at random in the range $[s\beta - 0.3s\beta, s\beta + 0.3s\beta]$, where β is the travel distance and s is the travel speed, which is set to 60km/h. Finally, the travel distance is computed as the Haversine distance, which is used as the distance measure in the WCP and to compute the variable travel costs in the PDP and SNDP.

The number of commodities to be transported between warehouse locations is given by $K \in \{N, 2N, 4N\}$; however for $N = 25$, only instances where $K \in \{25, 50\}$ are used. Each commodity k has an integer load q_k, which is selected uniformly at random from the set $[5, 20]$. As explained in Section 3, the time windows are indirectly assigned to the warehouse locations through the commodities. For the pickup time window W_k^p, e_k^p is selected uniformly at random from the business hours on day 1, and $l_k^p = e_k^p + \eta$ where η is an integer number of hours selected from the range $[2, 18]$. The delivery could occur on the second day, so for the delivery time window W_k^d, e_k^d is selected uniformly at random from the business hours across both days, i.e. [6:00, 22:00]∪[6:00 + 1, 22:00 + 1]. The end of the delivery time window is set to $l_k^d = e_k^d + \eta$, where η is defined above. However, it is imposed that $l_k^d \geq (l_k^p +$ travel time between pick-up and delivery $(tt_{\hat{o}_k^r \hat{d}_k^r})$ + load time $(g_{\hat{o}_k^r})$ + unloading time $(g_{\hat{d}_k^r}))$.

The instances are generated with a homogeneous set of vehicles. Each vehicle has a capacity of $U = Q^v = 100$ units and a total operation time of $T^v = 10$ hours. The use of each vehicle incurs a fixed cost of $f = 6000$, and the cost of directly delivering commodities is $\xi = 3f$. The PDP solved in the iterative algorithm accounts for the loading and unloading times of the commodities, which is set to 30 and 10 minutes, respectively, for all locations and commodities.

An instance for the SCSNDP is identified by the tuple (N, K, C), which indicates the number of warehouses, commodities and subregions. For each unique tuple using the data above, 5 different instances are randomly generated to diversify the test set. As such, a total of 80 instances are used in the computational experiments.

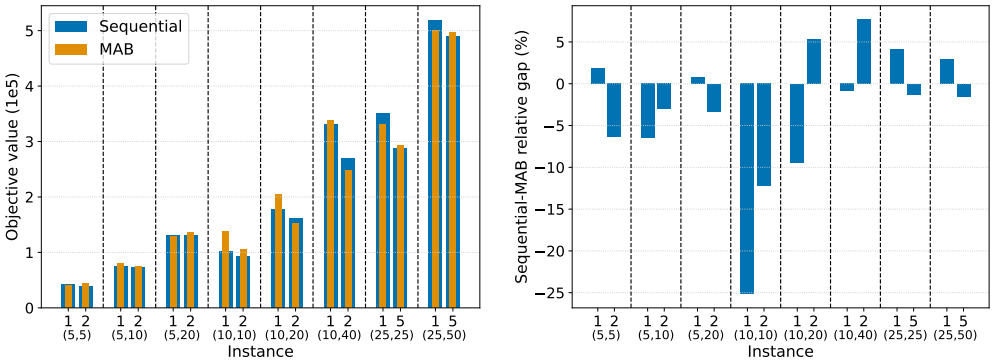
A maximum run time for all algorithms is set to 7200 seconds. The results are aggregated using an arithmetic mean over the 5 instances generated for the tuple (N, K, C). In all cases, times are reported in seconds.

5.2 Parallel SRIS settings

Two different settings are used for the parallel SRIS. The default setting, labelled *Default*, is described in Section 4.4. Briefly, *Default* executes a racing phase until sufficient executions of the iterative heuristic have been performed, then the parallel phase is executed, concurrently solving the relaxation and executing the iterative heuristic. An alternative setting, labelled *Racing Only*, executes the racing phase until the time limit is reached or the relaxation is solved by all solvers. *Racing Only* evaluates the performance of using the parallel architecture to execute a portfolio solver for the SCSNDP.

5.3 Comparing the sequential SRIS and MAB algorithm

The objective values for the best solutions found by the sequential SRIS and MAB algorithms are presented in Figure 9(a). It can be observed that across all collections of test instances, the sequential SRIS achieves an objective value that is better than the MAB algorithm in most cases. Specifically, the sequential SRIS achieves a better objective value in 10 of the 16 test instance collections. The magnitude of the improvement is captured in Figure 9(b). In the best case, the sequential SRIS achieves an objective value that is more than 25% better than the MAB algorithm (the (10, 10, 1) instance collection). In comparison, the best improvement achieved by the MAB algorithm over the sequential SRIS is 7.69% (the (10, 40, 2) instance collection). These results demonstrate that using the branch-and-bound search to guide the



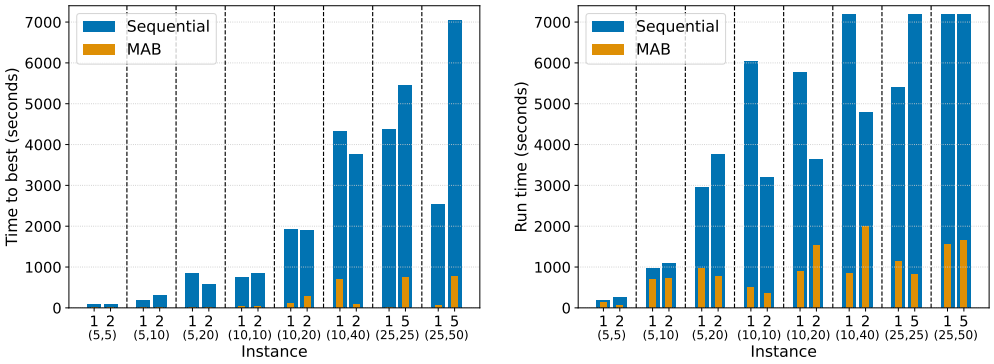
(a) Best objective value (b) Sequential-MAB relative gap

Figure 9: Objective values and relative gap of the best solutions found by the sequential SRIS and MAB algorithms. The relative gap is $(Sequential - MAB)/\max\{Sequential, MAB\}$.

iterative heuristic helps to find higher quality feasible solutions to the SCSNDP compared to using an MAB approach.

In Figure 9, it can be seen that for the larger instances from the test set, the MAB algorithm reports a better performance than the sequential SRIS. This is due to the fact that for these instances, finding a feasible solution, and improving that solution, is difficult for both algorithms. In many of these instances, the MAB algorithm is unable to improve upon the first feasible solution. It appears that the first feasible solution found by the MAB algorithm is quite good. Thus, in practice it would be valuable to find the first feasible solution by executing just one iteration of the MAB algorithm and then continuing with the sequential SRIS.

While the sequential SRIS achieves better quality feasible solutions compared to the MAB algorithm, this comes with a higher computational cost. Figure 10 presents the time until the best solution is found and total run time for the sequential SRIS and MAB algorithms. Considering the time to best solution, Figure 10(a), the sequential SRIS requires significantly more time than the MAB algorithm. This difference becomes more pronounced as the problem size increases. However, this increase in time to best solution can be explained by the fact that the iterative heuristic is only a subproblem of the sequential SRIS, as opposed to the core algorithm for the MAB algorithm. The sequential SRIS must balance the computational effort between executing the iterative heuristic and solving the relaxation. As a result, less time is available to search for feasible solutions in the sequential SRIS—compared to the MAB algorithm—leading to a longer time to find the best solution. However, it must be noted that the longer run times for the sequential SRIS lead to improved solutions to the SCSNDP in most



(a) Time to best (b) Run time

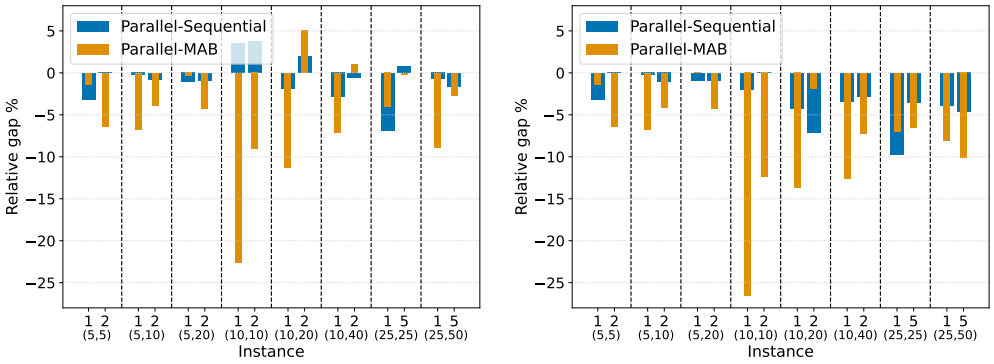
Figure 10: Time to best solution and total run time for the sequential SRIS and MAB algorithms.

instance collections evaluated.

Balancing of computational effort in the sequential SRIS also explains the significant difference in the run time of the sequential SRIS and MAB algorithms, as shown in Figure 10(b). Since the sequential SRIS solves the SCSNDP relaxation, the algorithm terminates only when the optimal solution to the relaxation is found. This differs significantly from the MAB algorithm, which has a number of termination criteria to limit the run time (see Hosoda et al. (2022) for more details). The additional time devoted to the sequential SRIS, compared to the MAB algorithm, is spent providing quality guarantees on the feasible solutions to the SCSNDP.

5.4 Evaluating the parallel SRIS

The performance of the sequential SRIS has demonstrated that using a branch-and-bound search to guide the iterative heuristic helps to find high quality feasible solutions. The parallel SRIS enhances the performance of the sequential SRIS and achieves a further improvement in the objective values for the SCSNDP. Figure 11 presents the relative difference between the parallel SRIS and both the sequential SRIS and MAB algorithms. Importantly, the results presented in Figure 11(b) show that *Racing Only* is able to achieve better solutions than both the sequential SRIS and MAB algorithms. This highlights the value of using a portfolio solver to find high quality solutions to the SCSNDP. By contrast, Figure 11(a) shows that *Default* still achieves an improvement in the objective value across most instance collections, but this is not as good as *Racing Only*. Overall, these results show that the parallel SRIS is effective in finding high quality solutions to the SCSNDP.

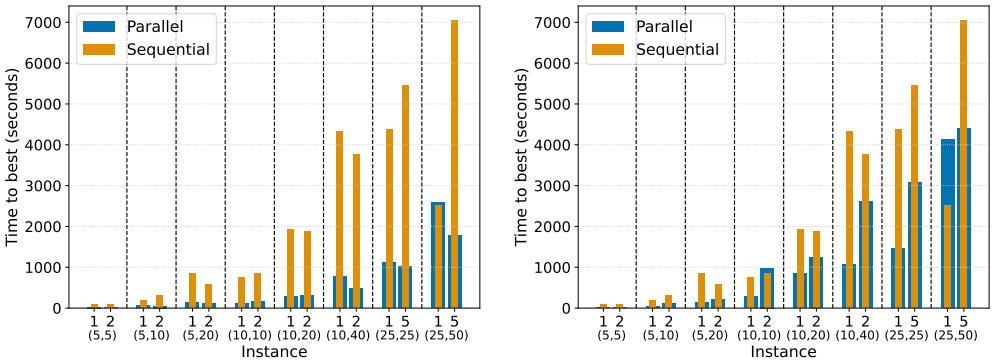


(a) *Default* (b) *Racing Only*

Figure 11: Comparing the objective of the best solutions found by the parallel SRIS against the sequential SRIS and MAB algorithms. The relative gap is $(Parallel - x)/\max\{Parallel, x\}$.

Since the parallel SRIS executes the sequential SRIS with different settings, it is expected that the former will always improve upon the latter. This is exemplified in Figure 11(b), where the sequential SRIS is run until optimality or the time limit is hit, but with multiple settings. By contrast, for *Default* it is not guaranteed that the setting used for the comparative sequential SRIS will be selected after the racing mode completes. Since only three settings are selected, it is possible that from those selected settings it will not find a solution better than the sequential SRIS. The results in Figure 11(a) indicate that a better winning solver selection scheme could further improve the performance of the parallel SRIS.

The value of the parallel SRIS is further highlighted when comparing the time until the best solution is found against the sequential SRIS. The results presented in Figure 12 show that the parallel SRIS is able to find high quality solutions to the SCSNDP faster than the sequential



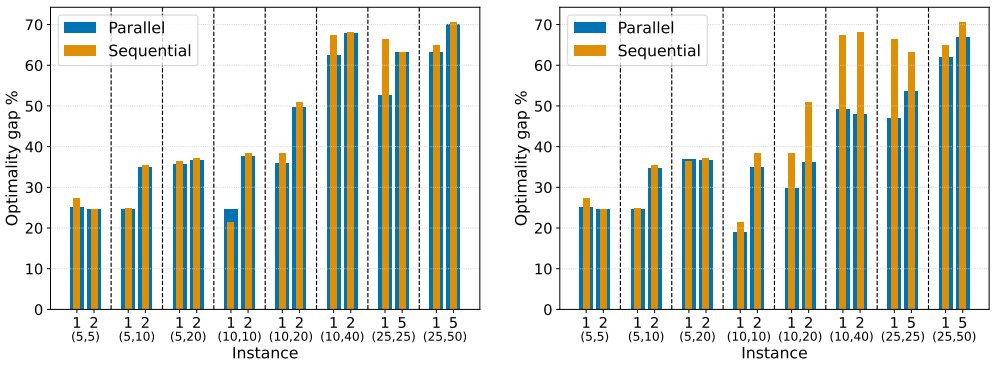
(a) *Default* (b) *Racing Only*

Figure 12: Comparing the time until the best solution for the parallel and sequential SRIS.

SRIS. Most notably, the time to best solution for *Default*, Figure 12(a), is significantly shorter than the sequential SRIS in all except 1 instance collection. However, for the exception, Figure 11(a) shows that the solution quality is much lower for the sequential SRIS. Comparing Figures 10(a) and 12(a), the time to best solution using *Default* is comparable with the MAB algorithm. While *Racing Only* requires more time to find the best solution, comparing Figures 11(b) and 12(b), this additional time leads to a significant improvement in the solution quality. Overall, the parallel SRIS is able to find better solutions for the SCSNDP faster than the sequential SRIS.

5.5 Evaluating solution quality

The use of the relaxation in solution algorithms for the SCSNDP aims to both guide the search for primal feasible solutions and provide quality guarantees on those solutions. Figure 13 presents the optimality gaps achieved by the parallel and sequential SRIS. The optimality gaps across the complete test set range from 25% up to 70%. While these gaps are quite large, it is the first time that quality guarantees are available for solutions to the SCSNDP. This result shows that more work is needed to improve the relaxation formulation and the quality of the lower bounds.

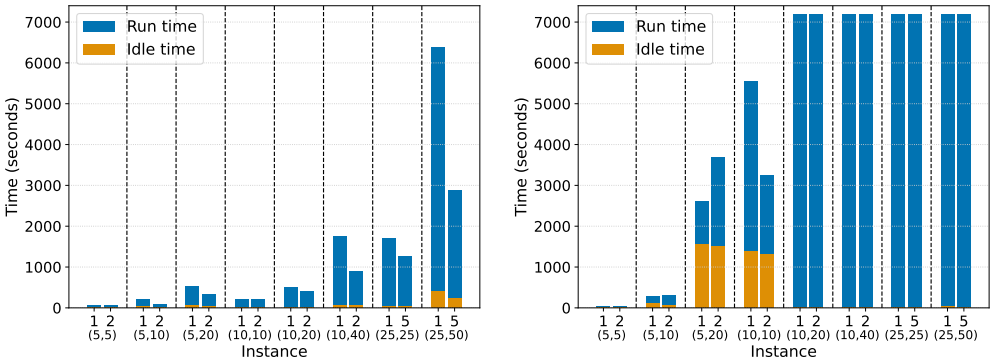


(a) *Default* (b) *Racing Only*

Figure 13: The relative optimality gaps achieved by the parallel and sequential SRIS. The optimality gap is $(Upper - Lower)/Upper$.

5.6 Parallel performance

An important aspect of parallel algorithms is the effective use of computational resources. A measure of this effectiveness is the idle time of solvers during the algorithm execution. For



(a) *Default* (b) *Racing Only*

Figure 14: The run time and idle time of the parallel SRIS.

this paper, the idle time is defined as the time that 1 or more solvers are inactive during the execution of the algorithm. The results comparing the idle time with the total run time are presented in Figure 14.

One striking observation from Figure 14 is that the total run time for *Default* is significantly less than for *Racing Only*. In particular, *Racing Only* exceeds the time limit for half of the instance collections, where *Default* did not exceed the time limit for any collection. Comparing Figures 11 and 14, there is a clear trade-off between the quality of solution and the execution time of the solution algorithm.

In regards to the idle time, the parallel SRIS exhibits very little idle time across the complete test set. For *Default*, shown in Figure 14(a), the largest proportion of idle time (22.87%) is observed for the (5, 5, 1) instance collection. While this is a high proportion of idle time, this is the result of very short run times for these instances—64.5 seconds on average. When considering the complete test set, *Default* achieves an average idle time of 8.23%. In comparison, Figure 14(b) shows that *Racing Only* exhibits a large amount of idle time for the instances that solve within the time limit—instance collections (5, 5, C), (5, 10, C), (5, 20, C) and (10, 10, C). For these instances, the smallest and largest proportion of idle time is 20.31% and 59.49% respectively. Similar to *Default*, this is partly driven by the very small run times to solve these instances; however, the (5, 20, C) and (10, 10, C) instance collections exhibit high absolute values of idle times. This is due to the nature of the racing algorithm, where it requires all solvers to terminate with the optimal solution to the relaxation. Thus, if a solver requires much longer run time than the others, this can drive high levels of idle time. The benefit of using the racing-only mode of the parallel SRIS is seen in the instances that did not solve within the

time limit. For these instances, the proportion of idle time is much less than 1%. Thus, during the racing phase of the parallel SRIS, all computing resources are being used to their maximum effectiveness. Considering the improvement in the objective values, seen in Figure 11(b), this use of computational resources by *Racing Only* has a clear benefit.

6 Conclusion

The SCSNDP is a large and complex supply chain management problem involving decisions on many different levels. Integrating location clustering with long haul and local vehicle routing introduces challenges in both the modelling and solving of this supply chain management problem. This paper addresses both of these challenges with the mathematical formulation for a relaxation of the SCSNDP and an effective parallel branch-and-bound based heuristic. The formulation developed in this paper is the first time an exact mathematical representation of this complex problem has been proposed. Building upon the work of Hosoda et al. (2022), the exact approach, solving the relaxation as a MIP, is combined with an iterative heuristic in a flexible parallelisation framework.

The proposed parallel heuristic is a major contribution of this paper, which is both effective for finding high quality solutions to the SCSNDP and flexible in its application to mathematical programming problems. Both a sequential and parallel version of the SRIS have been developed to exploit the flexibility in the UG framework. The computational results show that the sequential SRIS outperforms the MAB algorithm developed by Hosoda et al. (2022) in regards to solution quality. Further, combined with the solution to the relaxation, for the first time we are able to compute bounds on the best found solutions to the SCSNDP. The parallel SRIS is shown to enhance the performance of the SRIS by finding higher quality solutions in less time. As a major result from this work, the time to best solution when using the parallel framework is comparable to the MAB algorithm, but with the former achieving a higher solution quality.

This paper represents a major step forward in the solving of the SCSNDP. This is the first time an exact approach has been proposed, and as such, it is the first time that bounds on integer feasible solutions have been identified. As such, a clear direction of future work is to strengthen the bounds achieved by the relaxation and reduce the solution run times. While these are potentially contradicting research directions, achieving them will lead to both improved primal solutions and valid bounds. Another important direction of future research is

to improve the primal heuristics for the SCSNDP. The iterative algorithm is an adaptation of the MAB algorithm proposed by Hosoda et al. (2022). Alternative heuristics, or improvements to the iterative heuristic, can lead to lower cost solutions for the project partner. The combination of exact and heuristic methods have been shown to be very successful in improving our ability to solve the SCSNDP. Further improvements to the parallel framework and relaxation will drive us closer to solving this complex supply chain management problem to optimality.

References

Sérgio Barreto, Carlos Ferreira, José Paixao, and Beatriz Sousa Santos. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179 (3):968–977, 2007.

Natashia Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The continuous-time service network design problem. *Operations Research*, 65(5):1303–1321, 2017.

Paul Buijs, Iris F.A. Vis, and Héctor J. Carlo. Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, 239(3): 593–608, 2014.

Claudio Contardo, Vera Hemmelmayr, and Teodor Gabriel Crainic. Lower and upper bounds for the two-echelon capacitated location-routing problem. *Computers & Operations Research*, 39(12):3185–3199, 2012.

Ivan Contreras and Elena Fernández. General network design: A unified view of combined location and network design problems. *European Journal of Operational Research*, 219(3): 680–697, 2012.

Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius M Solomon, and François Soumis. VRP with pickup and delivery. *The vehicle routing problem*, 9:225–242, 2002.

Michael Drexl. Synchronization in vehicle routing—A survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

Michael Drexl and Michael Schneider. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241(2):283–308, 2015.

Gurobi. Gurobi - the fastest solver. See `https://www.gurobi.com/`. Last accessed: 13-01-2023.

Hilde Heggen, Yves Molenbruch, An Caris, and Kris Braekers. Intermodal container routing: Integrating long-haul routing and local drayage decisions. *Sustainability*, 11(6), 2019.

Kaj Holmberg and Di Yuan. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–481, 2000.

Junko Hosoda, Stephen J. Maher, Yuji Shinano, and Jonas Christoffer Villumsen. Location, transshipment and routing: An adaptive transportation network integrating long-haul and local vehicle routing. *EURO Journal on Transportation and Logistics*, 11:100091, 2022.

Marco Lam and John Mittenthal. Capacitated hierarchical clustering heuristic for multi depot location-routing problems. *International Journal of Logistics Research and Applications*, 16 (5):433–444, 2013.

Juliette Medina, Mike Hewitt, Fabien Lehuédé, and Olivier Péton. Integrating long-haul and local transportation planning: The service network design and routing problem. *EURO Journal on Transportation and Logistics*, 8(2):119–145, 2019.

Sanjay Melkote and Mark S Daskin. An integrated model of facility location and transportation network design. *Transportation Research Part A: Policy and Practice*, 35(6):515–538, 2001.

Snežana Mitrović-Minić and Gilbert Laporte. The pickup and delivery problem with time windows and transshipment. *INFOR: Information Systems and Operational Research*, 44(3): 217–227, 2006.

Gábor Nagy and Saïd Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.

Guido Perboli, Roberto Tadei, and Daniele Vigo. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380, 2011.

Hanne L. Petersen and Stefan Røpke. The pickup and delivery problem with cross-docking opportunity. In Jürgen W. Böse, Hao Hu, Carlos Jahn, Xiaoning Shi, Robert Stahlbock,

and Stefan Voß, editors, *Computational Logistics*, pages 101–113, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-24264-9.

Stefan Poikonen, Bruce Golden, and Edward A. Wasil. A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346, 2019.

Abdur Rais, Filipe Alvelos, and Maria do Sameiro Carvalho. New mixed integer-programming model for the pickup-and-delivery problem with transshipment. *European Journal of Operational Research*, 235(3):530–539, 2014.

Mohamed Salama and Sharan Srinivas. Joint optimization of customer location clustering and drone-based routing for last-mile deliveries. *Transportation Research Part C: Emerging Technologies*, 114:620–642, 2020.

Fernando Afonso Santos, Geraldo Robson Mateus, and Alexandre Salles da Cunha. The pickup and delivery problem with cross-docking. *Computers & Operations Research*, 40(4):1085–1093, 2013.

M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

Yuji Shinano, Stefan Heinz, Stefan Vigerske, and Michael Winkler. FiberSCIP—a shared memory parallelization of SCIP. *INFORMS Journal on Computing*, 30(1):11–30, 2018.

Nariaki Tateiwa, Yuji Shinano, Keiichiro Yamamura, Akihiro Yoshida, Shizuo Kaji, Masaya Yasuda, and Katsuki Fujisawa. CMAP-LAP: Configurable massively parallel solver for lattice problems. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 42–52, 2021.

UG version 1.0. UG: Ubiquity Generator framework. see `https://ug.zib.de/doc-1.0.0/html/`. Last accessed: 13-01-2023.

Yong Wang, Kevin Assogba, Yong Liu, Xiaolei Ma, Maozeng Xu, and Yinhai Wang. Two-echelon location-routing optimization with time windows based on customer clustering. *Expert Systems with Applications*, 104:244–260, 2018.

Min Wen, Jesper Larsen, Jens Clausen, Jean-François Cordeau, and Gilbert Laporte. Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60(12):1708–1718, 2009.

Nicole Wieberneit. Service network design for freight transportation: A review. *OR Spectrum*, 30(1):77–112, 2008.

David Wolfinger. A large neighborhood search for the pickup and delivery problem with time windows, split loads and transshipments. *Computers & Operations Research*, 126:105110, 2021.

David Wolfinger, Fabien Tricoire, and Karl F Doerner. A matheuristic for a multimodal long haul routing problem. *EURO Journal on Transportation and Logistics*, 8(4):397–433, 2019.