Subproblem Separation in Logic-Based Benders' Decomposition for the Vehicle Routing Problem with Local Congestion

Aigerim Saken
Department of Mathematics, University of Exeter, United Kingdom

Stephen J. Maher
Quantagonia GmbH, Bad Homburg, Germany

Abstract

Subproblem separation is a common strategy for the acceleration of the logic-based Benders' decomposition (LBBD). However, it has only been applied to problems with an inherently separable subproblem structure. This paper proposes a new method to separate the subproblem using the connected components algorithm. The subproblem separation is applied to the vehicle routing problem with local congestion (VRPLC). Accordingly, new Benders' cuts are derived for the new subproblem formulation. The computational experiments evaluate the effectiveness of subproblem separation for different methods applying new cuts. It is shown that subproblem separation significantly benefits the LBBD scheme.

2012 ACM Subject Classification Theory of computation → Mathematical optimization; Applied computing → Transportation

Keywords and phrases logic-based Benders' decomposition, vehicle routing, subproblem separation, connected components

Digital Object Identifier 10.4230/OASIcs.ATMOS.2023.16

Supplementary Material
Software (Source Code): `https://git.exeter.ac.uk/as1392/subproblem-separation-in-lbbd`

Acknowledgements Computational experiments were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) and National Academic Infrastructure for Supercomputing in Sweden (NAISS).

1 Introduction

Logic-based Benders' decomposition (LBBD) is an extension of classical Benders' decomposition. LBBD extends the classical Benders' approach by allowing the subproblem to be an optimisation problem of any form. LBBD was first introduced in Hooker [4] and then formalised in Hooker and Ottosson [7]. It generates Benders' cuts by using logical deductions from subproblem solutions, therefore it can handle any type of subproblem. This makes LBBD an effective method of combining different approaches such as mixed-integer programming (MIP) and constraint programming (CP).

Successful applications of LBBD include resource allocation and scheduling problems ([5],[12],[2],[19],[3],[9]), vehicle routing problems ([17],[14],[15],[10]), and other types of large-scale optmisation problems ([6]). Various acceleration techniques have been used in these applications in order to improve the effectiveness of the LBBD scheme. The common acceleration techniques are subproblem relaxation, cut strengthening, and subproblem separation. In their works on the evaluation of cut-strengthening techniques in LBBD, Saken et al. [18] and Karlsson and Rönnberg [8] show that for problems, which have inherently separable subproblems, benefits of applying subproblem separation dominate the benefits of applying cut strengthening.

Subproblem separation can be applied to problems with a bordered block-diagonal structure. This structure allows splitting the subproblem when master problem variables are fixed. We propose a new method for subproblem separation for problems that do not have such structure. On the example of the VRPLC, we demonstrate why problems with no block-diagonal structure can not be separated at the problem formulation stage. We then introduce a new method that applies the connected components algorithm to identify separable blocks of the subproblem based on the master problem solution in each iteration of the Benders' algorithm. The computational evaluation of the new method confirms that subproblem separation we propose has a significant impact on the LBBD scheme.

The rest of this paper is structured as follows. Section 2 describes the VRPLC. Section 3 presents the problem formulation. A brief description of the LBBD scheme for the problem is given in Section 4. The main contributions of this paper are Sections 5–7. Section 5 introduces the new method for subproblem separation. The derivation of new Benders' cuts is given in Section 6. The computational results in Section 7 demonstrate the effectiveness of the subproblem separation by evaluating the solution run times for the various new methods. Finally, concluding comments are given in Section 8.

Our contribution

The main contributions are:

- A new method of subproblem separation in the LBBD scheme using the connected components in a graph.
- A derivation and analytical validation of various Benders'cuts.
- Detailed computational experiments evaluating four different methods applying subproblem separation and the default method with no subproblem separation.
- The code related to the LBBD scheme and the subproblem separation is freely available at `https://git.exeter.ac.uk/as1392/subproblem-separation-in-lbbd`.

2 Vehicle routing problem with local congestion

The vehicle routing problem (VRP) is a problem that finds a set of suitable routes for a fleet of vehicles that deliver/collect goods from the central depot to a set of customers. The obtained routes must satisfy all of the customer requests while minimising the total travel distance and/or other costs. Some examples of practical applications are grocery delivery, parcel delivery, farm produce collection, waste collection etc. The VRP is one of the most extensively studied problems in optimisation due to both practical and theoretical interest.

There is no standard VRP formulation. However, one of the best-studied formulations of the vehicle routing problem is the capacitated VRP (CVRP)([16],[13],[20],[1],[17],[14],[15], and references therein). A lot of VRP problem formulations are based on CVRP. A feasible solution of CVRP is a set of routes starting and ending at the central depot. Every customer is visited only once on a specific route, and the cumulative demand (weight) of all requests the vehicle delivers must not exceed its capacity.

The VRPLC is the CVRP enriched with time window and local congestion constraints. This variant of the CVRP was introduced in Lam and Van Hentenryck [11]. The customer requests are grouped by locations. The congestion constraint at each location is a cumulative resource constraint that limits the number of vehicles present and/or in service at any given time. If all resources at a location are engaged, incoming vehicles must wait until the resources become available. This leads to time dependencies, and, subsequently, a scheduling substructure that is not present in conventional CVRPs [11]. An example of such time

Table 1 Data and decision variables for the model.

Name	Description
$T \in \{1, \ldots, \infty\}$	Time horizon
$l \in \{1, \ldots, \mathcal{L}\}$	Set of locations
n	Number of requests
$R = \{1, \ldots, n\}$	Set of requests
$C_l \in \{1, \ldots, \infty\}$	Resource capacity of location $l \in \mathcal{L}$
$R_l = \{i \in R \mid l_i = l\}$	Set of requests at location $l \in \mathcal{L}$
$l_i \in \mathcal{L}$	Location of request $i \in R$
$r_i \in [0, T]$	Release time of request $i \in R$
$d_i \in [0, T]$	Deadline of request $i \in R$
$p_i \in [0, T]$	Processing time of request $i \in R$
$q_i \in [1, Q]$	Weight of request $i \in R$
$Q \in \{0, \ldots, \infty\}$	Maximum weight a vehicle can carry
O^+	Artificial start that corresponds to the central depot
O^-	Artificial end that corresponds to the central depot
$\mathcal{N} = R \cup \{O^-, O^+\}$	Set of nodes
$\mathcal{A} = \{(i, j) \in \mathcal{N} \times \mathcal{N} \mid i \neq j\}$	Set of arcs connecting the nodes
c_{ij}	Travel time along arc (i, j)
$x_{ij} \in \{0, 1\}$	Indicates if a vehicle travels along arc (i, j)
$y_i^{\text{start}} \in [r_i, d_i]$	Time a vehicle starts unloading goods
$y_i^{\text{weight}} \in [q_i, Q]$	Total accumulated weight of delivered goods

dependency is that a delay on one route entails delays on other routes visiting the same location. These delays can cause infeasibility of a solution because of the time window constraints.

Developing a hybrid MIP and CP solver "Nutmeg", Lam et al. [10] introduce a logic-based Benders' decomposition (LBBD) scheme for the VRPLC. As VRPLC can be decomposed into a routing and a scheduling problem, it is naturally suited to LBBD. The authors use the branch-and-cut style of LBBD known as branch-and-check. Two objective types are considered – minimising total travel distance and minimising makespan.

3 Problem formulation

The problem formulation studied in this paper is based on the VRPLC formulation in Lam et al. [10]. However, we only consider the minimising total tardiness objective. The requests are allowed to be delivered past their time window, the tardiness of each request is the amount of time that has passed since the end of the window until the delivery time. The total tardiness is the sum of the tardiness of all requests.

The problem is to create a set of routes for vehicles to deliver goods from a central depot to various locations. The vehicles and the locations are subject to vehicle capacity and congestion constraints, respectively.

Table 1 lists the data and decision variables for the problem. The requests for goods are grouped by locations. Each request $i \in R$ must be delivered to location $l_i \in \mathcal{L}$ within a time window $[r_i, d_i]$, and all vehicles must return to the central depot before time T. Each

vehicle requires the use of one piece of equipment for processing time p_i to unload the goods for request $i \in R$. Each location only has the total fixed set of equipment C_l, the limited capacity of equipment then leads to location congestion.

The problem can be modeled using a graph $(\mathcal{N}, \mathcal{A})$. The central depot and each request $i \in R$ with the corresponding location information are represented through the set of nodes $\mathcal{N}$. The set $\mathcal{A}$ denotes the arcs connecting the nodes. The variables x_{ij} equal 1 if a vehicle travels along arc $(i, j) \in \mathcal{A}$, and 0 otherwise. Moving along arc (i, j) takes c_{ij} time units. There are two continuous subproblem variables at each node $i \in \mathcal{N}$. The continuous variables y_i^{start} and y_i^{weight} are equal to the time a vehicle starts unloading goods and the total accumulated weight of delivered goods, respectively.

The model for the vehicle routing problem with location congestion is given by

$$\min \sum_{i \in R} \max\{y_i^{\text{start}} + p_i - d_i, 0\} \tag{1}$$

$$\text{s.t.} \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad j \in R, \tag{2}$$

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad i \in R, \tag{3}$$

$$\textsc{Cumulative}((y_i^{\text{start}} | i \in R_l), (p_i | i \in R_l), (1 | i \in R_l), C_l), \quad l \in \mathcal{L}, \tag{4}$$

$$x_{ij} = 1 \rightarrow y_i^{\text{weight}} + q_j \leq y_j^{\text{weight}}, \quad (i, j) \in \mathcal{A}, \tag{5}$$

$$x_{ij} = 1 \rightarrow y_i^{\text{start}} + p_i + c_{ij} \leq y_j^{\text{start}}, \quad (i, j) \in \mathcal{A}, \tag{6}$$

$$x_{ij} \in \{0, 1\}, \quad (i, j) \in \mathcal{A}, \tag{7}$$

$$y_i^{\text{start}} \in [r_i, d_i], \quad i \in \mathcal{N}, \tag{8}$$

$$y_i^{\text{weight}} \in [q_i, Q], \quad i \in \mathcal{N}. \tag{9}$$

The objective function (1) minimises total tardiness of all requests. Constraints (2)–(3) ensure each node has exactly one incoming and outgoing arc. This ensures each request is assigned to exactly one vehicle. The CUMULATIVE constraints (4) enforce processing capacity limit at each location. Vector $((1|i \in R_l))$ represents resource requirement for each request $i \in R_l$. The CUMULATIVE constraints require the following: $\sum_{i \in R_{lt}} 1 \leq C_l$ for all times t, where $R_{lt} = \{i | y_i^{\text{start}} \leq t < y_i^{\text{start}} + p_i\}$ is the set of requests being processed at time t. Constraints (5)–(6) are only enforced when the corresponding values of x_{ij} are equal to 1. Constraints (5) ensure that the total accumulated weight of delivered goods by a vehicle does not decrease after each delivered request. Constraints (6) ensure request processing time and minimum travel times are respected. Constraints (6) are sufficient to avoid cycles. All of the vehicles are assumed to be identical, and each node has one incoming and outgoing arc, therefore the vehicles are not presented explicitly. The number of arcs outgoing from (or incoming to) the central depot gives the number of vehicles used in a solution.

4 Logic-based Benders' decomposition for VRPLC

The VRPLC decomposes into routing and scheduling components. The variables x_{ij} are viewed as the complicating variables. Fixing variables x_{ij} to trial values leads to a scheduling subproblem. The scheduling subproblem can be solved as a CP. The trial values of x_{ij} are found by solving the routing master problem as a MIP. The master problem identifies a set of vehicle routes that satisfy all delivery requests.

Let T denote the total tardiness. The master problem in iteration k is given by

$$\min\ T \tag{10}$$

$$\text{s.t.} \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad j \in R, \tag{11}$$

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij} = 1, \quad i \in R, \tag{12}$$

$$T \geq B_{x^i}(x), \quad i = 1, ..., k-1, \tag{13}$$

$$\text{[Valid inequalities]}. \tag{14}$$

The [Valid inequalities] contain constraints (5)–(6), they are added to the master problem to retain some information about the subproblem. They state that for a vehicle that travels along arc (i, j) the accumulated weight of goods delivered after request j cannot be smaller than the accumulated weight after request i. Similarly, unloading of goods at node j cannot start before unloading at node i. Inequalities (13) are the Benders' cuts obtained by solving the subproblem for master problem solutions x^i in iterations $i = 1, \ldots, k-1$. The Benders' cuts ensure feasibility and optimality of the problem solution.

Let x^k be the master problem solution in iteration k, then the subproblem is given by

$$\min \sum_{i\in R} \max\{y_i^{\text{start}} + p_i - d_i, 0\} \tag{15}$$

$$\text{s.t. } \textsc{Cumulative}((y_i^{\text{start}}|i \in R_l), (p_i|i \in R_l), (1|i \in R_l), C_l), \quad l \in \mathcal{L}, \tag{16}$$

$$x_{ij}^k = 1 \rightarrow y_i^{\text{weight}} + q_j \leq y_j^{\text{weight}}, \quad (i,j) \in \mathcal{A}, \tag{17}$$

$$x_{ij}^k = 1 \rightarrow y_i^{\text{start}} + p_i + c_{ij} \leq y_j^{\text{start}}, \quad (i,j) \in \mathcal{A}. \tag{18}$$

The subproblem is solved to schedule deliveries on the routes identified by the master problem.

The solution procedure is an iterative process that iterates between solving the master problem and the subproblem. Let T^* and T_k^* denote the optimal objective value of the master problem and the subproblem, respectively. In each iteration, the optimal value T^* provides a lower bound on the optimal value of (1)–(9), and T_k^* provides an upper bound. The optimal value T^* increases monotonically, the subproblem value T_k^* can increase or decrease. The procedure terminates when $T^* = \min\{T_1^*, \ldots, T_k^*\}$.

The main idea of LBBD is to use T_k^* and the reasoning behind this solution to obtain a bounding function $B_{x^k}(x)$ that gives a valid lower bound on the optimal value of (1). The bounding function $B_{x^k}(x)$ should have following two properties.

▶ **Property 1.** *$B_{x^k}(x)$ provides a valid lower bound on (1) for any given $x \in D_x$, where D_x is the domain of x. That is, $T \geq B_{x^k}(x)$ for any feasible (x, y) in problem .*

▶ **Property 2.** $B_{x^k}(x^k) = T_k^*$.

It is convenient to regard T_k^* as having an infinite value if the subproblem is infeasible. By this assumption, a strong duality holds for the dual of the subproblem: the optimal value of the subproblem is always equal to the optimal value of its dual [7].

▶ **Theorem 1** ([5]). *If the bounding function $B_{x^k}(x)$ satisfies properties 1 and 2 in each iteration of the Benders algorithm, and the domain D_y of y is finite, the Benders algorithm converges to the optimal value of (problem) after finitely many steps.*

Let $\mathcal{J}_k = \{(i,j) \in \mathcal{A} | x_{ij}^k = 1\}$ be the set of arcs that were selected in the master problem solution in iteration k. If the subproblem is infeasible, a feasibility cut given by

$$\sum_{(i,j)\in\mathcal{J}_k} (1 - x_{ij}) \geq 1 \tag{19}$$

is generated. If the subproblem has an optimal solution with value T_k^*, an optimality cut $T \geq B_{x^k}(x)$ is generated. The cut is given by

$$T \geq T_k^* \big(1 - \sum_{(i,j)\in\mathcal{J}_k} (1 - x_{ij})\big). \tag{20}$$

The cut indicates that the total tardiness T will have a value of at least T_k^*, unless one of the arcs $(i,j) \in \mathcal{J}_k$ is removed from the route.

Both feasibility cuts (19) and optimality cuts (20) can be routinely strengthened by replacing $\mathcal{J}_k$ with a smaller subset $\mathcal{J}_k^{'} \subseteq \mathcal{J}_k$, if the subproblem corresponding to $\mathcal{J}_k^{'}$ gives a solution with the same objective value as the solution for $\mathcal{J}_k$.

5 Subproblem separation

Subproblem separation is a common strategy used to accelerate the LBBD scheme. It is especially useful when the subproblem is a scheduling problem. Given that scheduling problems are difficult to scale up, splitting one big subproblem into many small independent subproblems usually benefits the solution procedure.

The subproblem can be separated when the problem has a bordered block diagonal structure, where the master variables define the border. Therefore, fixing the master variables to trial values makes the blocks separable. Meaning that the subproblem can be decoupled into a separate subproblem for each such block, and the solution of any decoupled subproblem does not depend on solutions of other subproblems.

The VRPLC formulation does not exhibit a block diagonal structure. Fixing master variables for the vehicle routing problem, does not naturally decouple the subproblem. One might consider decoupling the subproblem by each location. However, since a vehicle can travel through more than one location, constraints (5)–(6) create a border that connects the locations. For example, if a vehicle first travels through location l_1 and then location l_2, the subproblem for location l_2 would require the subproblem solution for l_1. This issue could be resolved by solving the subproblem for l_1 before solving subproblem for l_2. However, since a location may host several requests, it is possible that another vehicle travels through l_1 and l_2 in the opposite order, thus making this method of subproblem separation inapplicable. Therefore, the subproblem cannot be decoupled by locations at the problem formulation stage.

We propose to separate the subproblem during the solution process. One can note that some master subproblem solutions give routes that only connect some of the locations. For example, see Figure 1a, locations 1 and 4 are connected to each other and the central depot, while locations 2 and 3 are only connected to the central depot. One subproblem can be solved for locations 1 and 4 together, and one subproblem for each of locations 3 and 4.

We propose to identify separable blocks of locations in each iteration of the Benders' algorithm. The master problem solution x_{ij}^k can be represented as a graph $\mathcal{J}_k$. The edges in graph $\mathcal{J}_k$ are the connections between requests. In order to identify separable locations, a new graph is formed where the nodes are given by locations. The edges between requests are mapped onto the edges between locations. Connected locations can then be found by using

(a) Routes in the MP solution. **(b)** Connected components.

Figure 1 Example of a graph representing the central depot (D) and four locations.

an algorithm to identify connected components in the location graph. Our implementation uses a depth-first search (DFS) to identify connected components. Note that the edges that connect the central depot to the locations are ignored, otherwise, all of the locations will belong to a single connected component through the central depot. The number of independent calls of the DFS function is equal to the number of connected components. In the example above, (see Figure 1b) the connected components are $\{[1,4],[2],[3]\}$. A separate subproblem is then solved for each connected component. Observe that solving the subproblem when a connected component is a path is trivial.

It is important to note that the proposed algorithm can identify connected components for more general cases than the example above. Since a vehicle can travel to multiple locations, the algorithm ensures that all of the locations visited by one vehicle belong to a single component. Moreover, if several vehicles deliver requests to the same location, all of the locations traversed by the vehicles will be encompassed in a single component. This is possible due to the step of mapping the edges between requests to the edges between locations – several request nodes become one location node.

6 New Benders' cuts

There is an inherent computational benefit to splitting the subproblem into smaller independent subproblems. Nevertheless, in order to fully exploit the new subproblem structure it is important to generate strong Bender's cuts. In this section, we analytically derive different sets of valid Benders' cuts.

Let x^k be the master problem solution in iteration k, and let set $\mathcal{J}_k$ be given by $\mathcal{J}_k = \{(i,j)|x_{ij}^k = 1\}$. As mentioned in Section 5, since $\mathcal{J}_k \subseteq \mathcal{A}$ is a graph, it can be separated into connected components. Let set C_k denote the set of connected components in iteration k. The connected components partition the set $\mathcal{J}_k$, i.e.,

$$\bigcup_{c \in C_k} \mathcal{J}_k^c = \mathcal{J}_k \quad \text{and} \quad \mathcal{J}_k^c \bigcap \mathcal{J}_k^{c'} = \emptyset \quad c, c' \in C_k, \;\; c \neq c',$$

where $\mathcal{J}_k^c \subseteq \mathcal{J}_k$ is the set of all edges from $\mathcal{J}_k$ in the connected component c.

Sets $\mathcal{J}_k^c$ partition $\mathcal{J}_k$, cut (20) therefore can be rewritten as

$$T \geq T_k^* \big(1 - \sum_{c \in C_k} \sum_{(i,j) \in \mathcal{J}_k^c} (1 - x_{ij})\big). \tag{21}$$

Each connected component $c \in C_k$ describes a separate subproblem. The optimal objective for subproblem c in iteration k is given by T_{ck}^*. Further, cut (21) can then be rewritten as the first cut we are proposing to generate

$$T \geq \sum_{c \in C_k} T_{ck}^* \big(1 - \sum_{(i,j) \in \mathcal{J}_k^c} (1 - x_{ij})\big). \tag{22}$$

The cut (22) can be seen as a summation of cuts of type (20) for each component $c \in C_k$. Note that cut strengthening can be applied to all cuts presented in this section unless otherwise stated.

The second valid set of cuts we propose to generate in iteration k is given by

$$T_{ck} \geq T^*_{ck}\big(1 - \sum_{(i,j)\in\mathcal{J}^c_k}(1 - x_{ij})\big), \quad c \in C_k, \tag{23}$$

$$T \geq \sum_{c\in C_k} T_{ck}. \tag{24}$$

The auxiliary variables T_{ck}, denoting total tardiness for each connected component, are added to the master problem.

We now looking at splitting the cuts further. The main idea is to look at the edges $(i,j) \in \mathcal{A}$ in the routes identified by the master problem. When the subproblem is decoupled, the corresponding edges are also decoupled. Tardiness incurred by a vehicle traveling along (i,j) is denoted by T_{ij}. Consider cut (20), replacing T with $T = \sum_{(i,j)\in\mathcal{A}} T_{ij}$ results in

$$\sum_{(i,j)\in\mathcal{A}} T_{ij} \geq T^*_k\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x_{ij})\big). \tag{25}$$

Since $(i,j) \in \mathcal{A} \setminus \mathcal{J}_k$ have no influence on the bound given by cut (25), we can replace $\mathcal{A}$ by $\mathcal{J}_k$ and rewrite the cut as

$$\sum_{(i,j)\in\mathcal{J}_k} T_{ij} \geq T^*_k\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x_{ij})\big), \tag{26}$$

▶ **Theorem 2.** *Cuts* (26) *will provide a valid set of cuts to solve the problem to optimality.*

Proof. Cuts (26) can be presented in the form of Benders' cuts $T \geq B_{x^k}(x)$. Where in iteration k

$$T = \sum_{(i,j)\in\mathcal{J}_k} T_{ij}, \quad \text{and} \quad B_{x^k}(x) = T^*_k\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x_{ij})\big)$$

According to Theorem 1 and Properties 1 and 2, if for any feasible solution (x,y) the total tardiness is bounded such that $T \geq B_{x^k}(x)$, and $B_{x^k}(x^k) = T^*_k$, the Benders' algorithm converges to the optimal value.

We start from proving that $T \geq B_{x^k}(x)$ for any feasible (x,y). Note that, trivially $T \geq \sum_{(i,j)\in\mathcal{J}_k} T_{ij}$.

Let m be an iteration of the Benders' algorithm, such that $m \neq k$. Let set $\mathcal{J}_m$ be defined as $\mathcal{J}_m = \{(i,j)|x^m_{ij} = 1\}$. There can be three cases: $\mathcal{J}_m$ is a subset of $\mathcal{J}_k$, $\mathcal{J}_m$ is a superset of $\mathcal{J}_k$, and the symmetrical set difference $\mathcal{J}_m \triangle \mathcal{J}_k$ is non-empty. In the third case, the indices in $\mathcal{J}_m \setminus \mathcal{J}_k$ do not influence the bound by the definition of the cut, the indices in set $\mathcal{J}_k \setminus \mathcal{J}_m$ correspond to variables $x^m_{ij} = 0$ in the cut and do not influence the bound. Therefore, it is sufficient to consider the former two cases:

- $\mathcal{J}_m \subseteq \mathcal{J}_k$. This gives $\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x^m_{ij})\big) \leq 0$, since $\exists(i,j) \in \mathcal{J}_k$, such that $x^m_{ij} = 0$. Therefore $T \geq T^*_k\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x_{ij})\big)$.
- $\mathcal{J}_k \subseteq \mathcal{J}_m$. This gives $\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x^m_{ij})\big) = 1$, since $\forall(i,j) \in \mathcal{J}_k$, $x^m_{ij} = 1$. Therefore $T \geq T^*_k\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x_{ij})\big)$, because $T \geq T^*_k$.

We now prove that $B_{x^k}(x^k) = T_k^*$ in any iteration k:

$$B_{x^k}(x^k) = T_k^*\big(1 - \sum_{(i,j)\in\mathcal{J}_k}(1 - x_{ij})\big) = T_k^*, \quad \text{since } \forall(i,j) \in \mathcal{J}_k, \quad x_{ij}^k = 1. \qquad \blacktriangleleft$$

The third set of cuts we propose to generate can be derived by splitting the edges in $\mathcal{J}_k$ by the connected components. Since sets $\mathcal{J}_k^c$ partition $\mathcal{J}_k$, cut (26) can be rewritten as the set of cuts

$$\sum_{(i,j)\in\mathcal{J}_k^c} T_{ij} \geq T_{ck}^*\big(1 - \sum_{(i,j)\in\mathcal{J}_k^c}(1 - x_{ij})\big), \quad c \in C_k \tag{27}$$

$$T \geq \sum_{c\in C_k}\sum_{(i,j)\in\mathcal{J}_k^c} T_{ij}. \tag{28}$$

▶ **Theorem 3.** *Cuts* (27)–(28) *will provide a valid set of cuts to solve the problem to optimality*

Proof. A logic similar to the proof of Theorem 2 can be applied here. Let T_{ck} be defined as $T_{ck} = \sum_{(i,j)\in\mathcal{J}_k^c} T_{ij}$. We first prove that $T_{ck} \geq T_{ck}^*\big(1 - \sum_{(i,j)\in\mathcal{J}_k^c}(1 - x_{ij})\big)$ for any feasible solution (x, y).

Let $\mathcal{J}_m$ be the master problem solution in iteration m. The set J_m can be split by the connected components obtained in iteration k such that $\bigcup_{c\in C_k}\mathcal{J}_m^c = \mathcal{J}_m$. Similar to the proof of Theorem 2, it is sufficient to consider the following two cases:

- $\mathcal{J}_m^c \subseteq \mathcal{J}_k^c$. This gives $\big(1 - \sum_{(i,j)\in\mathcal{J}_k^c}(1 - x_{ij}^m)\big) \leq 0$, since $\exists(i,j) \in \mathcal{J}_k^c$, such that $x_{ij}^m = 0$. Therefore $T_{ck} \geq T_{ck}^*\big(1 - \sum_{(i,j)\in\mathcal{J}_k^c}(1 - x_{ij})\big)$.
- $\mathcal{J}_k^c \subseteq \mathcal{J}_m$. This gives $\big(1 - \sum_{(i,j)\in\mathcal{J}_k^c}(1 - x_{ij}^m)\big) = 1$, since $\forall(i,j) \in \mathcal{J}_k^c$, $x_{ij}^m = 1$. Therefore $T_{ck} \geq T_{ck}^*\big(1 - \sum_{(i,j)\in\mathcal{J}_k^c}(1 - x_{ij})\big)$ $\blacktriangleleft$

7 Computational experiments

The computational effectiveness of subproblem separation and various Benders' cuts, described in Section 6, is evaluated in a series of computational experiments. The evaluated cuts are cuts (22), cuts (23)–(24), cuts (27), and the combination of cuts (23)–(24) and cuts (27). Using the combination of cuts (23)–(24) and cuts (27) means generating both sets of cuts in each iteration. Each experiment solves the VRPLC with the minimising total tardiness objective. The experiments run the LBBD scheme with the separated subproblem generating each type of cuts separately. Another experiment runs the default implementation of the LBBD scheme with no subproblem separation. The different implementations are referred to as "methods" for the sake of brevity. Since it is important to apply cut strengthening to accelerate the solution process, the deletion filter cut-strengthening technique ([8],[18]) has been applied in all computational runs.

The main metric of computational effectiveness is the impact of different methods on the run time of the LBBD scheme. For the run time plots, the horizontal axis gives the time and the vertical axis gives the percentage of solved instances. A point (x, y) on the curve means that y% of instances can be solved in less than x seconds.

Experiment setting

The LBBD scheme is implemented in Python 3.8, and the MIP and CP models are solved using Gurobi Optimizer version 9.1.2 and IBM ILOG CP Optimizer version 20.1, respectively. All tests have been carried out on a computer with two Intel Xeon Gold 6130 processors (16 cores, 2.1 GHz each) and 96 GB RAM. Each instance was given a total time of 20 minutes and the MIP-gaps are set to 0 for the master problems.

Instances

We use 450 instances from Lam et al. [10], the instances are available at `https://github.com/ed-lam/nutmeg/tree/master/examples/vrplc/Instances`. The instances are generated for 5, 8, or 10 locations. For each number of locations, there are instances with 20, 30, and 40 requests. Location resource capacities vary between one and eight for all instances The instances are modified for the minimising total tardiness objective with deadlines decreased by 10%.

Percentage of solved instances

The main observation from Figure 2 is that all of the methods applying subproblem separation outperform the default implementation. The default implementation with 12% of solved instances is notably behind the methods applying subproblem separation. Cuts (23)–(24) have the highest effectiveness with 81.5% of solved instances. Cuts (22) and the combination of cuts (23)–(24) and (27) have marginally lower percentages of solved instances – 77.8% and 80%, respectively.

An interesting result is that although cuts (27) outperform the default method, they show significantly lower results than other methods applying subproblem separation – 28.2% of solved instances. The main reason is that the cuts lead to repeated master problem solutions with the same connected components. This implies that this type of cut introduces symmetry that is difficult to handle for the master problem solver. The results for the combination of cuts (27) and cuts (23)–(24) being slightly lower than the results for cuts (23)–(24) also imply that cuts (27) adversely impact the run time.

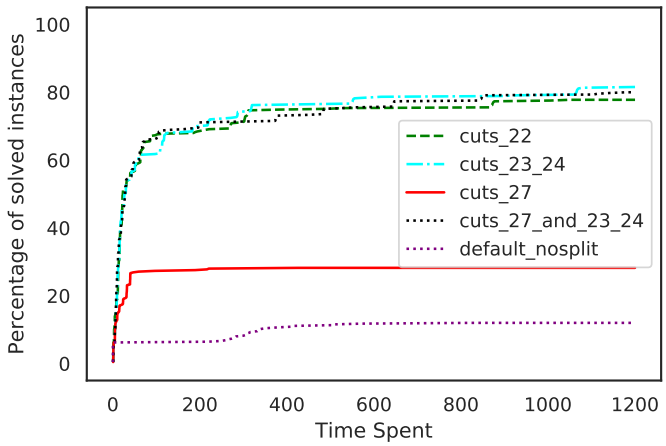

Figure 2 Percentage of instances solved to optimality for the minimising tardiness for cumulative scheduling problem.

Table 2 The table presents the average master problem solution time, average subproblem solution time, and number of subproblems solved per instance. The instances for which the results were not retrieved within 20 minutes are omitted.

Method	T_{MP}	T_{sub}	T_{sub} *(per iter)*	N_{sub}	N_{iter}	N_{inst}	N_{inst} *(solved)*
Default	1.36	3.75	0.67	1453	40.3	105	53
Cuts 22	0.92	0.138	0.02	430	11.25	352	349
Cuts 23-24	1.09	0.141	0.016	578	17.6	367	366
Cuts 27	0.26	0.065	0.03	108	3.3	126	126
Combination	1.19	0.068	0.015	502	12.11	361	359

The results in Table 2 highlight the effect of subproblem separation on the LBBD scheme. The reported values are calculated for different sets for each method. For each method, the set comprises retrieved instances that were either solved to optimality or timed out. The N_{inst} column indicates the number of instances in each set. The number of instances solved to optimality by each method is given in the last column. As can be seen in Table 2, the default implementation spends much more time solving subproblem per instance – 3.75 seconds compared to 0.141 seconds and below by other methods. This can be explained by the greater average subproblem solution time per Benders' iteration – 0.67 seconds compared to 0.02 seconds, 0.016 seconds, 0.03 seconds, and 0.015 seconds by cuts (22), cuts (23)–(24), cuts (27), and the combination of cuts (23)–(24) and cuts (27), respectively. This result shows that it takes less time to solve multiple smaller subproblems than to solve one subproblem. Another interesting observation is that the default implementation leads to a higher number of Benders' iterations and subproblems solved, this suggests that the cuts generated by the default implementation are less effective than the cuts generated by the other methods.

8 Conclusion

This paper proposes a new implementation of the LBBD scheme for the vehicle routing problem with local congestion. We propose using the connected components algorithm to identify separable blocks of the subproblem. The new implementation reformulates the separated subproblem in each Benders' algorithm iteration. This method of separating the subproblem can be applied to other vehicle routing problems with vehicle capacity and congestion constraints. Since the new reformulation requires new Bender's cuts, we derive various types of cuts. We then evaluate subproblem separation and new Benders' cuts in computational experiments.

The main conclusion is that subproblem separation is an effective technique for accelerating the LBBD scheme for the vehicle routing problem with local congestion. However, in order to fully exploit the new subproblem structure, it is important to generate strong cuts. Splitting the cuts by the connected components showed the best computational results. Whereas, splitting the cuts by edges was not effective. An area of future work is to investigate methods to handle the difficulty introduces by these cuts.

References

1 Florian Arnold, Michel Gendreau, and Kenneth Sörensen. Efficiently solving very large-scale routing problems. *Computers & Operations Research*, 107:32–42, 2019.

2 Elvin Coban and John N Hooker. Single-facility scheduling by logic-based benders decomposition. *Annals of Operations Research*, 210:245–272, 2013.

3 Simon Emde, Lukas Polten, and Michel Gendreau. Logic-based benders decomposition for scheduling a batching machine. *Computers & Operations Research*, 113:104777, 2020.

4 John N Hooker. *Logic-Based Benders Decomposition*, chapter 19, pages 389–422. John Wiley & Sons, Ltd, 2000.

5 John N Hooker. Planning and scheduling by logic-based benders decomposition. *Operations research*, 55(3):588–602, 2007.

6 John N Hooker. Logic-Based Benders Decomposition for Large-Scale Optimization. *Large Scale Optimization in Supply Chains and Smart Manufacturing: Theory and Applications*, pages 1–26, 2019.

7 John N Hooker and Greger Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

8 Emil Karlsson and Elina Rönnberg. Strengthening of Feasibility Cuts in Logic-Based Benders Decomposition. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 18th International Conference, CPAIOR 2021, Vienna, Austria, July 5–8, 2021, Proceedings 18*, pages 45–61. Springer, 2021.

9 Emil Karlsson and Elina Rönnberg. Logic-based Benders decomposition with a partial assignment acceleration technique for avionics scheduling. *Computers & Operations Research*, 146:105916, 2022.

10 Edward Lam, Graeme Gange, Peter J Stuckey, Pascal Van Hentenryck, and Jip J Dekker. Nutmeg: a MIP and CP Hybrid Solver Using Branch-and-Check. *SN Operations Research Forum*, 1:1–27, 2020.

11 Edward Lam, Panos M. Pardalos, and Pascal Van Hentenryck. A branch-and-price-and-check model for the vehicle routing problem with location congestion. *Constraints*, 21:394–412, 2016.

12 Michele Lombardi and Michela Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17:51–85, 2012.

13 Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9:61–100, 2017.

14 Günther R Raidl, Thomas Baumhauer, and Bin Hu. Speeding up Logic-Based Benders' Decomposition by a Metaheuristic for a Bi-Level Capacitated Vehicle Routing Problem. In *International Workshop on Hybrid Metaheuristics*, pages 183–197. Springer, 2014.

15 Günther R Raidl, Thomas Baumhauer, and Bin Hu. Boosting an Exact Logic-Based Benders Decomposition Approach by Variable Neighborhood Search. *Electronic Notes in Discrete Mathematics*, 47:149–156, 2015.

16 Ted K. Ralphs, Leonid Kopman, William R. Pulleyblank, and Leslie E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94:343–359, 2003.

17 Sarmad Riazi, Carla Seatzu, Oskar Wigström, and Bengt Lennartson. Benders/gossip methods for heterogeneous multi-vehicle routing problems. In *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–6. IEEE, 2013.

18 Aigerim Saken, Emil Karlsson, Stephen J. Maher, and Elina Rönnberg. Computational Evaluation of Cut-Strengthening Techniques in Logic-Based Benders' Decomposition. *SN Operations Research Forum*, 4:62, 2023.

19 Defeng Sun, Lixin Tang, and Roberto Baldacci. A Benders decomposition-based framework for solving quay crane scheduling problems. *European Journal of Operational Research*, 273(2):504–515, 2019.

20 Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*, 257(3):845–858, 2017.