



Citation for published version:

Arslan, O, Kumcu, GÇ, Kara, BY & Laporte, G 2021, 'The location and location-routing problem for the refugee camp network design', *Transportation Research Part B: Methodological*, vol. 143, pp. 201-220.
<https://doi.org/10.1016/j.trb.2020.11.010>

DOI:

[10.1016/j.trb.2020.11.010](https://doi.org/10.1016/j.trb.2020.11.010)

Publication date:

2021

Document Version

Peer reviewed version

[Link to publication](#)

Publisher Rights

CC BY-NC-ND

University of Bath

Alternative formats

If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

The location and location-routing problem for the refugee camp network design

Okan Arslan^{a,*}, Gül Çulhan Kumcu^b, Bahar Yetiş Kara^c, Gilbert Laporte^{a,d}

^a*HEC Montréal and CIRRELT, Montréal, Canada*

^b*Rutgers Business School, Rutgers University, New Jersey, USA*

^c*Department of Industrial Engineering, Bilkent University, Ankara, Turkey*

^d*School of Management, University of Bath, Bath, United Kingdom*

Abstract

The refugee crisis is one of the major challenges of modern society. The influxes of refugees are usually sudden and the refugees are in sheer need of services such as health care, education and safety. Planning public services under an imminent humanitarian crisis requires simultaneous strategic and operational decisions. Inspired by a real-world problem that Red Crescent is facing in Southeast Turkey, we study the problem of locating refugee camps and planning transportation of public service providers from their institutions to the located camps. Our modeling approach brings a new facet to the location and routing problem by considering the location of beneficiaries as variables. We develop a branch-price-and-cut algorithm for the problem. To solve the pricing problem, we introduce a cycle-eliminating algorithm using nested recursion to generate elementary hop constrained shortest paths. The best version of our algorithm efficiently solves 244-node real-world instances optimally.

Keywords: refugee camp location, location-routing problem, vehicle routing problem, transportation, humanitarian logistics, branch-price-and-cut

1. Introduction

The world is now witnessing a massive refugee crisis. According to the United Nations High Commissioner for Refugees (UNHCR, 2019), the number of displaced persons is 68.5 million and more than 37% of these people are under the age of 18. One in every 110 people in the world is a refugee. The refugee crisis has significant socio-economic long-term impacts. Furthermore, the numbers are increasing at an alarming rate, especially since the 2011 Syrian refugee crisis (UNHCR, 2019).

Top five refugee-hosting countries as of 2019 are Turkey (3.5 million), Uganda (1.4 million), Pakistan (1.4 million), Lebanon (1.0 million), and Iran (0.98 million) (UNHCR, 2019), all of which are classified as ‘developing economies’ by the United Nations Department of Economic

*Corresponding author

Email address: okan.arslan@hec.ca (Okan Arslan)

11 & Social Affairs (UN/DESA, 2019). The role of the hosting country is central to the welfare of
12 the refugees. During a humanitarian crisis, massive influxes of people are usually sudden, and the
13 hosting countries face many strategic and operational level decisions in the aftermath. These coun-
14 tries are partially or fully responsible for opening and managing refugee camps for the displaced
15 people and providing the necessary needs and public services such as safety, health care and ed-
16 ucation. While some basic needs are covered by the facilities located within the camps, certain
17 additional services need to be provided by the local authorities (Karsu et al., 2019). Examples of
18 such services include specialist doctor or psychiatrist visits for the self-development of children
19 and youth refugees (UNICEF, 2011).

20 The UNHRC's operations in Turkey are among the largest worldwide (UNHCR, 2019). Our
21 research is inspired and shaped by a real-world problem faced by the Turkish Red Crescent in
22 southeast Turkey for opening camps and providing public services. The aim is to locate refugee
23 camps, select the service hosting institutions, and determine the routes of service providers from
24 the selected service hosting institutions to the selected camps. It is closely related to the classical
25 location-routing problems (see, e.g., Drexler and Schneider, 2017) and it entails the additional lo-
26 cation decision of the users, who are the refugee camps in our context. It has applications in the
27 public sector, where the user locations are also decisions to be made by the planning authorities,
28 mainly in disaster management after a major evacuation or during a major confinement. Examples
29 include temporary housing after natural disasters such as hurricanes, floods, wildfires or earth-
30 quakes (see, e.g., Sherali et al., 1991; Murray-Tuite and Wolshon, 2013). When the evacuees are
31 located in their temporary shelters, providing basic services such as health care and security is
32 crucial, and can be costly particularly when the duration of the temporary housing is long. An-
33 other application arises in managing the major confinements. For instance, during the COVID-19
34 pandemic, individuals arriving from abroad to Turkey were located for a period of two weeks in se-
35 lected student dormitories and the municipalities provided health care services and basic supplies
36 for these facilities (Anadolu Agency, 2020). We model the problem as a location and location-
37 routing problem (LLRP) and we develop efficient solution techniques for it.

38 *1.1. Literature review*

39 Humanitarian crises have natural as well as man-made origins. The United Nations has adopted
40 several measures to alleviate sufferings of refugees, namely those who are forced to leave their
41 home country in order to escape major conflict zones. Humanitarian logistics has emerged as a
42 research field and has garnered significant interest over the last decade, particularly in view of the
43 fact that nearly 75% of the funding allocated to disaster response is related to the supply chain
44 (Van Wassenhove, 2006). The field is rooted in the early works of Altay and Green (2006) and
45 Van Wassenhove (2006) and has since evolved into a rich research area. For surveys on humani-
46 tarian logistics, we refer the reader to Çelik et al. (2012), Galindo and Batta (2013), Özdamar and
47 Ertem (2015), Kara and Savaşer (2017), and Besiou and Van Wassenhove (2019).

48 Disaster operations are generally classified into four phases: mitigation, preparedness, re-
49 sponse and development (Çelik et al., 2012). Besiou and Van Wassenhove (2019) portray the
50 history of humanitarian logistics into three phases by investigating the past, present and future.
51 They note that the future of the field is driven by multiple factors, including the management of

52 large numbers of refugees, which lies at the heart of our study. We will focus on the location and
53 routing components of the problem.

54 Location-routing problems combine decisions from the tactical and operational levels. Recent
55 surveys are those of Prodhon and Prins (2014), Albareda-Sambola (2015) and Drexl and Schneider
56 (2017). In humanitarian logistics, location problems are typically solved in the mitigation and pre-
57 paredness phases (Balcik and Beamon, 2008; Paul and Wang, 2019), while routing problems occur
58 in the response phase (Campbell et al., 2008; Balcik et al., 2008; Huang et al., 2012, 2013; Sheu,
59 2014; Oruc and Kara, 2018). Examples include welfare operations to distribute perishable food to
60 welfare agencies in an equitable and efficient way (Eisenhandler and Tzur, 2019a,b) and dynamic
61 transportation planning for ambulances after an earthquake (Mills et al., 2018). The integration of
62 location and routing is rarely observed in the humanitarian logistics literature. One of the early
63 works is due to Yi and Özdamar (2007), who considered an integrated logistics support and evac-
64 uation operations in the disaster response phase. Balcik (2017) developed a tabu search algorithm
65 for site selection and vehicle routing for post-disaster rapid needs assessment. Cherkesly et al.
66 (2019) recently solved a location, covering and routing problem arising in design of a community
67 health care network in Liberia.

68 *1.2. Scientific contributions and organization of this paper*

69 The peculiarities of LLRP that set it apart from previous studies are twofold. First, the location
70 decision of the refugee camps as well as the routing decision among these camps are taken by
71 the same authority. In other words, the location decisions pertain to the depots and also to the
72 refugees who are the users. The second feature that differentiates LLRP from the existing location
73 and routing problems is the fact that the number of service providing depots is larger than the
74 number of candidate refugee camps.

75 The scientific contributions of this paper are as follows:

- 76 • We introduce the refugee camp location and public service planning problem.
- 77 • The problem constitutes a new facet of location-routing problems through the consideration
78 of user locations.
- 79 • We develop a path-based model and a specialized branch-price-and-cut (BP&C) exact algo-
80 rithm. We deal with complications arising from the additional location decisions. Particu-
81 larly, we develop a cycle-canceling algorithm to solve the pricing problem of the model.
- 82 • We assess the performance of our methodology on real-world datasets.

83 The remainder of this paper is organized as follows. We present the problem in Section 2
84 and devise a BP&C algorithm in Section 3. We present the computational results and discuss the
85 implications on real-world data derived from the Southeastern part of Turkey in Section 4. We
86 conclude the study in Section 5.

2. Formal Problem Definition

We now provide a formal definition of the LLRP. Consider a graph $G = (N, A)$ where N and A are the sets of nodes and arcs, respectively. Let d_{ij} be the distance from node $i \in N$ to node $j \in N$. The nodes represent the candidate refugee camps (or simply *camps*) and the service hosting institutions (or simply *hosts*). Let $R \subset N$ be the set of candidate camps. There are Q refugees to be located in the refugee camps, and each camp $i \in R$ has capacity q_i . Let f_i be the location cost of camp $i \in R$. As usual, all costs relate to the same planning horizon. Set S represents the public services planned by the government for the arriving refugees, such as health care and safety. Every open camp must be provided with all the services in the set S . For each service type $s \in S$, there is a set of hosting institutions $H_s \subset N$ where the service providers are located. For instance, for the health care service, hospitals represent the hosting institutions and the medical doctors and the nurses are the service providers. We define H as the set of all hosts, that is, $H = \bigcup_{s \in S} H_s$. Without loss of generality, a host is assumed to provide a single type of service. If this is not the case, a host node can be duplicated as many times as the number of services it provides. There are C_h service providers of type $s \in S$ located at host $h \in H_s$, which we refer to as the host capacity. Let L_h be the maximum number of camps that can be visited in a single trip by a service provider located at host $h \in H$. A service provider of type $s \in S$ departs from a host $h \in H_s$, visits at most L_h camps and returns back to host h ; we refer to such a path as a *feasible path*. The problem under consideration is formally defined as follows:

Definition 1. Consider a graph $G = (N, A)$, camps $R \subset N$, hosts $H \subset N$, services S and an integer Q . The LLRP is defined as the problem of selecting camps in the set R in order to locate Q refugees and finding a set of feasible paths in graph G to ensure that all types of services in S are provided for every open camp such that the camp and host capacities are respected and the total cost of opening camps and routing service vehicles is minimized.

3. Branch-price-and-cut Algorithm

In this section, we present a set-covering model and develop an exact BP&C decomposition algorithm, which has been shown as one of the best performing algorithms for vehicle routing and location-routing problem classes (Ozbaygin et al., 2017; Wang and Sheu, 2019; Ozbaygin and Savelsbergh, 2019). The novelties in the BP&C algorithm developed in this study are manifold.

- The pricing problem considered in this study is a hop-constrained shortest path problem. We develop a recursive cycle-eliminating algorithm in Section 3.4. This algorithm can be used to solve a LRP with the same pricing problem or separately in other problem types.
- The site-selection decisions require changing the branching rules, which are presented in Section 3.8.
- We use knapsack inequalities in Section 3.9 to strengthen the LP relaxation of the model.

122 *3.1. Path-Based model*

Let \mathcal{P} be the set of all feasible paths π , \mathcal{P}_i^s be the set of paths of service type $s \in S$ visiting camp $i \in R$, and $\overline{\mathcal{P}}_h$ be the set of paths $\pi \in \mathcal{P}$ starting and ending at host $h \in H$. In our formulation, the variable y_π equals 1 if path π is used and 0 otherwise, variable x_i equals 1 if camp $i \in R$ is selected and 0 otherwise and variable z_h equals the number of service providers used from host h . We refer to y_π , x_i and z_h as path, location and host variables, respectively. The parameter $a_{i\pi}$ is the number of times path π visits node $i \in N$ and d_π is the routing cost of path π in the planning horizon. The path-based model, which we refer to as PM, is:

$$(PM) \text{ minimize } \sum_{i \in R} f_i x_i + \sum_{\pi \in \mathcal{P}} d_\pi y_\pi \quad (1)$$

subject to

$$\sum_{i \in R} q_i x_i \geq Q \quad (2)$$

$$x_i \leq \sum_{\pi \in \mathcal{P}_i^s} a_{i\pi} y_\pi \quad i \in R, s \in S \quad (3)$$

$$\sum_{\pi \in \overline{\mathcal{P}}_h} y_\pi = z_h \quad h \in H \quad (4)$$

$$z_h \leq C_h \quad h \in H \quad (5)$$

$$x_i \in \{0, 1\} \quad i \in R \quad (6)$$

$$y_\pi \in \{0, 1\} \quad \pi \in \mathcal{P} \quad (7)$$

$$z_h \geq 0 \text{ and integer} \quad h \in H. \quad (8)$$

123 The objective function minimizes the total cost of opening camps and routing the service providers.
 124 Constraint (2) ensures that the total camp capacity is sufficient to host all the refugees. Constraints
 125 (3) imply that if a camp is selected, then all service types must be provided. When only elementary
 126 paths are considered, then $a_{i\pi} = 1$ for all $i \in R$ and $\pi \in \mathcal{P}$. In non-elementary paths, $a_{i\pi}$ can assume
 127 values greater than one, but the correctness of the model is still ensured (Desrochers et al., 1992).
 128 Note that constraints (3) link the location and routing variables. Setting $Q = \sum_{i \in R} q_i$ in constraint
 129 (2) requires $x_i = 1$ for all $i \in R$ and the model essentially solves a location-routing problem.
 130 Constraints (4) and (5) jointly enforce capacity limits for hosts. The auxiliary variable z_h is used in
 131 the formulation to enforce the branching rules in the branch-and-bound (B&B) tree. Constraints
 132 (6), (7) and (8) define the domains of the variables. This formulation is different from the classical
 133 set-covering formulation for the vehicle routing problem (Desrosiers and Lübbecke, 2005) due to
 134 the additional site-selection decisions associated with refugee camps, which introduces additional
 135 complexities in the branch-and-price (B&P) framework.

136 *3.2. Column generation*

137 The PM model has exponential number of paths variables, which makes it impossible to solve
 138 directly. We have therefore developed a B&P algorithm (Lübbecke and Desrosiers, 2005) as an
 139 exact solution methodology. A central component of this method is the column generation proce-
 140 dure used to solve the linear programming (LP) relaxation of the PM, which we refer to as PM-R.

141 In order to prevent having an upper bound on path variables in the LP relaxation of the model, we
 142 convert path variables into integer variables, without loss of generality.

We first have a restricted formulation with an initial set of path variables. We then solve this restricted LP and determine whether there exists any path variable with a negative reduced cost. The problem of finding such a variable is referred to as the pricing problem. **We solve this problem for every host to find a variable with a negative reduced cost, add such variables to the restricted problem and reiterate this process until no path variable with a negative reduced cost remains.** Let α_i and β_h be the dual variables associated with Constraints (3) and (4), respectively. The reduced cost of path variable y_π is equal to

$$c_\pi = d_\pi - \sum_{i \in N(\pi)} a_{i\pi} \alpha_i + \beta_{\pi_0} \quad \pi \in \mathcal{P}, \quad (9)$$

143 where $N(\pi)$ is the set of camps in path π and π_0 is the first node in the path, representing the origin
 144 host. Having a path π with $c_\pi < 0$ implies that y_π needs to be added to the formulation in the next
 145 iteration. To ensure that no such path exists, we need $\min_{\pi \in \mathcal{P}} \{c_\pi\} \geq 0$. Therefore, the pricing problem
 146 is to minimize $d_\pi - \sum_{i \in N(\pi)} a_{i\pi} \alpha_i + \beta_{\pi_0}$ over all feasible paths $\pi \in \mathcal{P}$ in the graph G .

147 3.3. Pricing problem

148 The pricing problem can be modeled as a resource-constrained shortest-path problem (RCSPP)
 149 on a graph with negative cost cycles (Irnich and Desaulniers, 2005). In this section, we use the
 150 terms ‘shortest path’ and ‘minimum cost path’ interchangeably to refer to a path with the minimum
 151 reduced cost. In the LLRP, the number of camps that can be visited in a path is the only resource
 152 and this reduces the pricing problem into the hop-constrained shortest path problem (HSPP) on a
 153 graph with negative cost cycles. To ensure the convergence of the column generation scheme, such
 154 a pricing problem needs to be solved for every host $h \in H$. When the solution path is required to
 155 be elementary, the HSPP is known to be NP-hard (Dahl and Gouveia, 2004).

156 For a given host $h \in H$, consider a graph $\hat{G}(h) = \hat{N}, \hat{A}$ where h' is a new dummy node to
 157 represent arrival to h , set $\hat{N} = R \cup \{h, h'\}$, \hat{A}_1 is the set of arc in G induced by nodes in \hat{N} , \hat{A}_2 is the
 158 set of new arcs (i, h') for all $i \in R$ with length d_{ih} and $\hat{A} = \hat{A}_1 \cup \hat{A}_2$. An HSPP instance is a quintuple
 159 $\langle h, L_h, \hat{G}(h), \alpha, \beta_h \rangle$ where all parameters are as previously defined. The objective of solving such
 160 an instance is to identify a minimum cost elementary path in $\hat{G}(h)$ from origin h to destination h' ,
 161 visiting at most L_h camps, where cost of a path is defined as in (9).

Let v_{ij} be a binary variable indicating whether arc (i, j) is traversed or not, and let r_i be a binary variable indicating whether camp $i \in R$ is visited or not. The HSPP can be modeled as follows:

$$\text{minimize } \sum_{(i,j) \in A} d_{ij} v_{ij} - \sum_{i \in R} \hat{\alpha}_i r_i + \hat{\beta}_h \quad (10)$$

subject to

$$\sum_{j:(i,j) \in A} v_{ij} - \sum_{j:(j,i) \in A} v_{ji} = \begin{cases} 1 & \text{if } i = h \\ -1 & \text{if } i = h' \\ 0 & \text{otherwise} \end{cases} \quad i \in N \quad (11)$$

$$\sum_{i \in R} r_i \leq L_h \quad (12)$$

$$r_i = \sum_{j: (i,j) \in A} v_{ij} \quad i \in R \quad (13)$$

$$\sum_{(i,j) \in S} v_{ij} \leq |S| - 1 \quad S \subset A : 2 \leq |S| \leq |N| - 1 \quad (14)$$

$$r_i, v_{ij} \in \{0, 1\} \quad i \in R, (i, j) \in A. \quad (15)$$

162 The objective function minimizes the reduced cost. Constraints (11) are the flow conservation
 163 constraints. Constraint (12) ensures that no more than L_h camps are visited on the path. If a camp
 164 $i \in R$ is visited, then the corresponding r_i variable is forced to be equal to one through constraints
 165 (13). Since there exists a trade-off between the r and v variables in the objective function, there
 166 may exist subtours in the solution. Constraints (14) prevent the existence of such subtours. Finally,
 167 constraints (15) define the domains of the variables.

168 Identifying an elementary hop-constrained shortest path using the preceding model is compu-
 169 tationally expensive. Therefore, we do not use it for solving the pricing problem. Instead, we
 170 develop a recursive algorithm in Sections 3.4 and 3.5 based on cycle elimination.

171 To accelerate the solution of the pricing problem, the elementarity requirements of the paths
 172 can be relaxed. The PM formulation is still valid under this assumption since a camp will only
 173 appear at most once in the optimal solution. Therefore, the optimal solution remains unchanged
 174 regardless of the elementarity assumption. The pricing problem is then transformed into non-
 175 elementary HSPP in graphs with negative cost cycles, which can be solved by the Bellman-Ford
 176 algorithm (Ford, 1956; Bellman, 1958) in $O(|N|^3)$ time. However, the LP relaxation of set covering
 177 models for vehicle routing problem types is tighter when the paths are cycle-free (Desrochers
 178 et al., 1992). Generally, eliminating 2-cycles (i, j, i) from the paths improves the bounds and
 179 boosts the performance of the algorithm significantly. In the following section, we present a cycle-
 180 eliminating algorithm. For brevity, we use the term ‘ σ -cycle elimination’ for a given integer σ to
 181 imply that all cycles of size not exceeding σ are eliminated from the paths.

182 3.4. A cycle-eliminating algorithm for non-elementary HSPP

183 The exact labeling algorithm developed in this section is used to find shortest paths without
 184 any σ -cycles for $\sigma \geq 2$. Consider an HSPP instance $\langle h, L_h, \hat{G}(h), \alpha, \beta_h \rangle$. In our algorithm, labels
 185 represent partial paths originating from h , visiting a number of camps no greater than L_h . A label
 186 ℓ is a vector of nodes (h, i_0, \dots, i_n) where h is the host node, i_0, \dots, i_{n-1} are camps and i_n is a camp
 187 if ℓ is a partial path and it is the dummy node h' when it is a full path. For convenience, we refer
 188 to a node on a partial path by its position relative to the last node of the partial path. The last node
 189 of ℓ is therefore represented as $\ell[-1]$. Other nodes are referred accordingly. The reduced cost of
 190 label ℓ is $c(\ell)$. We use the sign \oplus for concatenating partial paths. We also refer to all outgoing and
 191 incoming arcs of node i as δ_i^+ and δ_i^- , respectively.

192 An extension of a label is the operation of adding a new node to the end of the partial path,
 193 adjacent to the last node. At each iteration of our algorithm, all labels are extended once and we
 194 refer to such an iteration as a *pass*. In each pass, the first step is the extension of current labels to
 195 create new σ -cycle-free labels. In the second step, dominance rules are applied to keep the labels

196 that have the potential to be an optimal path. In our algorithm, we determine the non-dominated
 197 labels to guarantee optimality and we discard all other labels since they are dominated. For each
 198 σ , the dominance rules change. To determine non-dominated labels for σ -cycle elimination, we
 199 need the following sequence of results. We first start with dominance rule for 2-cycle elimination.

200 **Proposition 1.** *Consider a host h , node i and associated set of labels \mathcal{L}_i . A label $\ell_1 \in \mathcal{L}_i$ is
 201 non-dominated in order to eliminate 2-cycles if and only if there exists a node $j \in \delta_i^+$ such that
 202 $j \neq \ell_1[-1]$ and the partial path $\ell_1 \oplus \{j\}$ has the minimum reduced cost among all partial paths
 203 $\ell_2 \oplus \{j\}$ with $j \neq \ell_2[-1]$ and $\ell_2 \in \mathcal{L}_i$.*

204 **PROOF.** (Necessity) Assume that there exists such a node $j \in \delta_i^+$. When $j \neq \ell_1[-1]$, the partial path
 205 $\ell_1 \cup \{j\}$ does not create a 2-cycle and is therefore feasible. Since the partial path created by this
 206 extension has the minimum reduced cost among other possible extensions, it has the potential of
 207 having the minimum reduced cost, which implies that it is not dominated.

208 (Sufficiency) By contraposition, assume that for all nodes $j \in \delta_i^+$ such that $j \neq \ell_1[-1]$, the
 209 partial path $\ell_1 \cup \{j\}$ does not have the minimum reduced cost among all partial paths created by
 210 extension of other labels. This implies that for every node $j \in \delta_i^+$ such that $j \neq \ell_1[-1]$, there exists
 211 another label $\ell_j \in \mathcal{L}_i$, for which $\ell_j \cup \{j\}$ has the minimum reduced cost among among all partial
 212 paths $\ell_2 \cup \{j\}$ with $j \neq \ell_2[-1]$ and $\ell_2 \in \mathcal{L}_i$. Clearly, for every possible extension, ℓ_1 is dominated.
 213 \square

214 Note that, in order to eliminate 2-cycles, the condition for a label to be non-dominated is satis-
 215 fied by investigating the costs when extending this label in the next pass. This reasoning can be
 216 generalized and can be extended to σ -cycle elimination by considering the following $\sigma - 1$ passes.

217 **Proposition 2.** *Consider a host h , node i and associated set of labels \mathcal{L}_i . A label $\ell_1 \in \mathcal{L}_i$ is non-
 218 dominated in order to eliminate σ -cycles if and only if there exists a partial path $pPath$, with at
 219 most σ nodes, such that $(\ell_1 \oplus pPath)$ does not create a σ -cycle and $(\ell_1 \oplus pPath)$ has the minimum
 220 reduced cost among all partial paths $(\ell_2 \oplus pPath)$ that does not contain a σ -cycle and $\ell_2 \in \mathcal{L}_i$.*

221 The proof is very similar to that of Proposition 1 and we omit for conciseness. Algorithm 1 is
 222 the cycle-eliminating HSPP algorithm, which we refer to as CHSPP. Forbidden nodes and arcs
 223 can also be imposed in the algorithm, which is needed in the B&B tree for fixing variables and
 224 for implementing branching decisions (as will become clear in the following section). For a given
 225 graph \hat{G} , a host h , a hop limit L_h , a cycle-elimination limit σ , sets of forbidden nodes and arcs
 226 $fNodes$ and $fArcs$, and dual values $\alpha_i, i \in R$, and β_h , the algorithm outputs a minimum cost path
 227 from h to h' without any cycles of size σ or less, using at most L_h camps and not using any of
 228 the nodes in $fNodes$ or arcs in $fArcs$. We now describe the algorithm. In the initialization phase,
 229 empty label sets are created for all the nodes in \hat{N} and a single label is added to the source node
 230 with a zero reduced cost. In the Bellman-Ford algorithm, the main loop is executed $|N|$ times so
 231 that after $|N|$ updates either a cycle is detected or the shortest path to the destination is found.
 232 Similarly, the main loop of our algorithm is executed $L_h + 1$ times since a feasible path can have
 233 at most L_h camps. We first create temporary label sets in lines 6–7 of Algorithm 1 to keep the
 234 new potential labels. Those that do not contain a σ -cycle are then recorded in the labels set in

235 lines 15–16. In the label extension phase described in lines 8–14, every label in node $i \in \hat{N}$ is
 236 extended to all nodes adjacent to i , for all $i \in \hat{N}$. Once all updates are completed, we identify the
 237 non-dominated labels in lines 17–21. Labels are first sorted with respect to their cost in line 18.
 238 An empty set *newLabelsSet* is created in line 19 to keep the list of non-dominated labels, and a
 239 set of nodes is created in line 20 to keep track of possible future extensions, which corresponds
 240 to *pPath* in Proposition 2. A nested loop in line 21 determines the non-dominated nodes. The
 241 function *nestedLoop* checks the conditions presented in Proposition 2. It starts by eliminating
 242 2-cycles (level 2) and continues by incrementing the parameter *level* one by one until it equals
 243 σ . When canceling σ cycles, the number of iterations between the current pass and $L_h + 1$ can
 244 potentially be less than σ . For this reason, $\min\{L_h + 1 - pass, \sigma\}$ future passes are checked in
 245 line 21. Note that if a label has the same cost as the minimum cost, it is also non-dominated.
 246 Such labels are identified in lines 33–37. They can provide alternative optimal solutions, which is
 247 needed when the algorithm is customized to return multiple paths instead of only a single path.

248 3.5. A labeling algorithm for elementary HSPP

249 We now show that a feasible path is elementary if it does not include any σ -cycles with $\sigma =$
 250 $L_h - 1$.

251 **Proposition 3.** *If a path (h, i_1, \dots, i_n, h') with $n \leq L_h$ does not contain any σ -cycles with $\sigma =$
 252 $(n - 1)$, then it is elementary.*

253 **PROOF.** Node h has only outgoing arcs and h' has only incoming arcs. Furthermore, these two
 254 nodes representing departure and arrival from and to a host do not appear in the set of intermediate
 255 nodes. The longest possible cycle then has length $L_h - 1$ when $i_1 = i_{L_h}$. \square

256 Therefore, when $\sigma = L_h - 1$, Algorithm 1 generates an elementary hop constrained shortest
 257 path.

258 3.6. A heuristic algorithm for elementary HSPP

259 The most time-consuming step of Algorithm 1 is lines 17–21, where the *nestedLoop* function
 260 is called. Instead of this costly step, we apply a heuristic which consists of keeping only the top
 261 σ labels generally finds a path with negative reduced cost, if one exists. If this heuristic cannot
 262 identify a path with negative reduced cost path, the exact algorithm is then applied.

263 3.7. Initial set of columns

264 The column generation procedure requires an initial set of path variables, which is needed to
 265 generate the dual variables. Even though a feasible solution of the model does not require to have
 266 a path from every host, we still need to have at least one path variable originating from each host
 267 in the initial pool of path variables in order to generate all the dual variables. Similarly, a feasible
 268 solution does not require visiting all camps, but we need to have variables visiting every camp
 269 in the initial pool of path variables so that the corresponding dual variables can be obtained, and
 270 the reduced costs in the next iteration of the column generation can be calculated correctly. To
 271 this end, the selection of initial set of columns is crucial. In our implementation, we use a greedy
 272 heuristic to generate an initial set of path variables starting from each host h and visiting the closest
 273 camp until a maximum of L_h camps have been visited. This process is reiterated until all camps
 274 are visited.

Algorithm 1: Cycle-eliminating hop-constrained shortest path algorithm (CHSPP)

Input: $\hat{G} = (\hat{N}, \hat{A}), h, L_h, \sigma, fNodes, fArcs, \beta_h, \alpha_i \quad i \in R$.

Output: A hop-constrained σ -cycle-free path with the minimum reduced cost

```
1 Function Main():
2   foreach  $i \in \hat{N} \setminus fNodes$  do // Initialization
3      $labels[i] \leftarrow EmptyLabelsSet()$ ;
4    $labels[h].add(\{h\}, 0)$ ;
5   for  $pass = 1$  to  $L_h + 1$  do // Main loop
6     foreach  $i \in \hat{N} \setminus fNodes$  do
7        $tempLabels[i] \leftarrow EmptyLabelsSet()$ ;
8     foreach  $i \in \hat{N} \setminus fNodes$  do // Extension of Labels
9       foreach  $\ell \in labels[i]$  do
10        foreach  $j \in \delta^+(i) : (i, j) \notin fArcs$  do
11          if  $j$  is not contained in the last  $\sigma$  nodes of  $\ell$  then // Cycle-canceling
12            if  $i == h'$  then  $dualCost \leftarrow \beta_h$ ; else  $dualCost \leftarrow -\alpha_j$ ;
13             $newLabel \leftarrow newLabel(\ell \oplus \{j\}, c(\ell) + d_{ij} + dualCost)$ ;
14             $tempLabels.add(newLabel)$ ;
15        foreach  $i \in \hat{N} \setminus fNodes$  do // Update
16           $Labels[i].add(tempLabels[i])$ ;
17        foreach  $i \in \hat{N} \setminus fNodes$  do // Domination
18           $sortedLabels[i] \leftarrow labels[i]$  sorted in ascending order with respect to cost;
19           $newLabelsSet \leftarrow EmptyLabelsSet()$ ;
20           $nodes \leftarrow newIntegersSet[\sigma]$ ;
21           $labels[i] \leftarrow nestedLoop(nodes, i, newLabelsSet, sortedLabels, 2, \min\{L_h + 1 - pass, \sigma\})$ ;
22  return path constructed from the minimum cost label in  $labels[h']$ .
```

```
23 Function nestedLoop(nodes, i, inputSet, sortedLabels, level,  $\sigma$ ):
24   if  $level > \sigma$  then return inputSet;
25   foreach  $j \in \delta^+(i) : (i, j) \notin fArcs$  do
26      $nodes[level] \leftarrow j$ ;
27     if  $fNodes.contains(node[level])$  then continue;
28     outerWhileLoop: while sortedLabels has label do
29       Pop  $\ell$  from sortedLabels;
30        $futurePath \leftarrow (nodes[2], \dots, nodes[level])$ ;
31       if  $(\ell.path \oplus futurePath)$  does not contain an  $m$ -cycle for  $m \leq \sigma$  then
32          $inputSet.add(\ell)$ ;
33         while sortedLabels has label do
34           Pop  $\ell_2$  from sortedLabels;
35           if  $c(\ell_2) > c(\ell)$  then break outerWhileLoop;
36           if  $(\ell_2 \oplus futurePath)$  does not contain an  $m$ -cycle for  $m \leq \sigma$  then
37              $inputSet.add(\ell_2)$ ;
38       if  $level \neq \sigma$  then nestedLoop(nodes, i, inputSet, sortedLabels, level + 1,  $\sigma$ );
39  return inputSet
```

275 3.8. Branch-and-price for an integer solution

276 Once we have the solution of the PM-R by column generation, we apply branching to obtain
277 an integer feasible solution. There are binary location variables, integer host variables and integer
278 path variables in PM. We use a three stage hierarchical branching rule and we always branch on
279 the most fractional variable in all three levels of branching rules. We first branch on the location
280 variables x_i . The branching rules on x_i are enforced by adding a single constraint in the master
281 problem. The pricing problem does not change with this branching rule since we forbid node i
282 in the pricing problem only if $x_i = 0$. If all the location variables are binary, we then branch
283 on the host variables z_h . A single constraint is added to enforce the branching rules, which does
284 not change the pricing problem. In the third level, we consider the path variables y_π . Using the
285 standard branching rule on path variables creates two unbalanced branches: one forcing the path
286 to be selected, and the other branch corresponding to the path not being selected. Selecting a path
287 is a strong decision; however, eliminating a path from the formulation is weak since alternative
288 paths similar to the eliminated one can easily be found in dense graphs. Furthermore, forbidding a
289 path in the pricing problem is not trivial (Feillet, 2010). For these reasons, branching on (implicit)
290 arc variables has been widely used in B&P algorithms in vehicle routing contexts. When each
291 customer is visited only once, enforcing (implicit) arc variables in a compact formulation implies
292 the use of binary path variables. However, in our context, an arc can potentially be traveled
293 by multiple vehicles providing different services. This necessitates having integer arc variables,
294 which is not straightforward from an implementation point of view. Our solution for enforcing
295 the path variables to binary values is by branching on an implicit variable corresponding to arc-
296 host combinations. Let r_{ij}^h be a binary variable equal to 1 if and only if a path originates from
297 host h and visits arc (i, j) . Branching on r_{ij}^h either forces an arc to be traveled by a vehicle from
298 host h or forces it not to be used from this host. We branch on the arc-host combination that has
299 the most fractional arc flow. Note that the second and third level branching rules ensure that the
300 y_π variables are binary, as these two branching rules has widely been applied in vehicle routing
301 context (Bettinelli et al., 2011).

302 Another complication of the problem at hand is the fact that there is no requirement to visit
303 every potential camp, but only selected ones. Therefore a careful treatment is needed for correct
304 branching implementation. Forbidding arcs and nodes is used at both the PM-R level and at the
305 pricing problem level to enforce the branching decisions.

306 3.8.1. Branching 1 (Forcing host h to use arc (i, j)):

307 To enforce this rule at the PM-R level, all variables associated with paths starting at host h and
308 using an arc emanating from node i or entering into node j , other than arc (i, j) , are removed from
309 the formulation. Let s be the service of host h . All variables associated with paths starting at a host
310 $\hat{h} \in H^s \setminus \{h\}$ and using arc (i, j) are also removed from PM-R. These removals ensure that if nodes
311 i and j are visited, then arc (i, j) is the only option. However, this is still not sufficient to ensure
312 that arc (i, j) is traveled by a vehicle starting at host h , because the camps are not required to be
313 visited in the problem. Therefore, we also need to set $x_i = x_j = 1$ to ensure that arc (i, j) appears
314 in the solution. A similar technique is implemented at the pricing problem level. When solving
315 the pricing problem for host h , we forbid all arcs leaving node i and entering node j , except arc
316 (i, j) . When solving the pricing problem for hosts $\hat{h} \in H^s \setminus \{h\}$, we forbid nodes i and j . This only

317 leaves the possibility of using arc (i, j) by a path starting at host h . Note that, unlike the PM-R
 318 level operation, we do not force arc (i, j) to be used at the pricing problem level. Forcing arc (i, j)
 319 in the pricing problem can possibly lead to missing negative-length paths.

320 3.8.2. Branching 2 (Forbidding arc (i, j) for host h):

321 We remove all variables corresponding to paths starting from host h and using arc (i, j) from
 322 PM-R. To ensure that no such paths are generated by the pricing problem, we forbid arc (i, j)
 323 when solving the pricing problem for host h . Our primary goal in branching is to improve the
 324 lower bound. To this end, we select the node with the best bound in the B&B tree.

325 3.9. Cutting planes

326 When no paths are generated by the pricing algorithm, the solution of the LP relaxation can
 327 be separated to improve the lower bound by adding violated inequalities. One such inequality
 328 is the knapsack cover inequality (Wolsey, 1975) for constraint (2). A set C is called a cover
 329 if $\sum_{i \in C} q_i > \sum_{i \in R} q_i - Q$, or equivalently $Q > \sum_{i \in R \setminus C} q_i$. For all such covers, the knapsack cover
 330 inequality $\sum_{i \in C} x_i \geq 1$ is valid for PM-R. Given an LP solution \hat{x} , a violation can be determined if
 331 $\gamma = \min_{C \subset N} \left\{ \sum_{j \in C} \hat{x}_j \mid \sum_{j \in R \setminus C} q_j < Q \right\} < 1$, which requires solving another optimization problem. Instead,
 332 we use a greedy heuristic to determine a violated knapsack inequality. We first sort the camps
 333 in ascending order according to \hat{x} . Starting from the top, we add camps to the cover set C until
 334 $\sum_{i \in C} q_i > \sum_{i \in R} q_i - Q$, at which point we check whether $\sum_{i \in C} \hat{x}_i < 1$. This implies that $\sum_{i \in C} x_i \geq 1$ is
 335 violated and we add this inequality and reiterate to generate new columns if possible.

336 3.10. Acceleration techniques

337 We now discuss two methods to accelerate the implementation to produce improved upper and
 338 lower bounds.

339 3.10.1. Variable Fixing:

340 In vehicle routing problems, variable fixing by reduced costs is generally applied on the im-
 341 plicit arc variables (Costa et al., 2019). However, the reduced costs of the implicit variables are
 342 not directly available in a path-based model. On the other hand, PM contains location variables
 343 that are strongly linked with the implicit arc variables. Setting a location variable equal to zero
 344 disqualifies all paths using the corresponding node. Furthermore, the reduced costs of the location
 345 variables are available at every node of the of the B&B tree. Let \bar{z} and \underline{z} be the upper and lower
 346 bounds on the optimal value of the PM, respectively, and let χ_i be the reduced cost of location
 347 variable x_i for all $i \in N$. A location variable x_i , $i \in N$ can then be fixed to zero if $\chi_i > \bar{z} - \underline{z}$. At
 348 PM the level, a constraint is added to set the corresponding location variable equal to zero. At the
 349 pricing problem level, a node is ‘forbidden’ and therefore no path using the corresponding node is
 350 generated.



Figure 1: Map of Turkey (cities with refugee camp operations in the Southeastern part are highlighted in gray color)

351 *3.10.2. Heuristics:*

352 After solving the root node relaxation of the B&B tree, we solve an integer programming model
 353 with the columns so far generated to possibly find a good upper bound before starting to branch
 354 (Rothenbacher et al., 2018). We also call this integer programming model **whenever one hundred**
 355 **or more new variables have been generated** in the B&B tree and before reaching the time limit. All
 356 such models aim to generate good upper bounds and are limited in time. In our implementation,
 357 we run this heuristic for at most two minutes.

358 **4. Computational Study**

359 We have implemented the algorithms using Java and CPLEX 12.9.0.0, and all experiments
 360 were conducted on a cluster of 26 machines each with two Intel Xeon X5675 3.07-GHz proces-
 361 sors with 96 GB of RAM running on Linux. Each machine has 12 cores, and each experiment was
 362 run using a single thread. The time limit for all experiments was set to six hours. In the follow-
 363 ing, we describe the dataset, study the impacts of path elementarity on the LP relaxation of our
 364 model, determine the algorithm parameters, present the computational experiments, and analyze
 365 the results on a real-world dataset.

366 *4.1. Data*

367 Our dataset originates from the field in the southeast Turkey, where refugee camps are located
 368 along the Turkish-Syrian border. Figure 1 highlights the 10 cities in the region of interest: Adana,
 369 Adıyaman, Gaziantep, Hatay, Kahramanmaraş, Kilis, Malatya, Mardin, Osmaniye, and Şanlıurfa.

370 Turkey’s Law No. 6458 on Foreigners and International Protection entitles refugees to health
 371 care, education and social services. To this end, we consider these three types of services to be
 372 provided to the refugees. The corresponding hosting institutions are hospitals, high schools and
 373 municipality city halls. There are 60 hospitals, 77 high schools and 33 municipality city halls in
 374 the region. A set of 74 locations are identified as potential refugee camps that are either currently
 375 serving as camps or that are non-residential large areas suitable for the establishment of a refugee

Network	No. cities	No. candidate refugee camps	No. hospitals	No. high schools	No. municipality city halls	Nb. Nodes
Network 1	3	18	3	6	3	30
Network 2	3	18	6	12	7	43
Network 3	3	18	15	28	8	69
Network 4	5	41	6	9	5	61
Network 5	5	41	12	20	13	86
Network 6	5	41	26	42	14	123
Network 7	7	41	7	13	7	68
Network 8	7	41	15	29	19	104
Network 9	7	41	44	54	25	164
Network 10	10	74	11	19	10	114
Network 11	10	74	23	43	27	167
Network 12	10	74	60	77	33	244

Table 1: Characteristics of the networks

camp. Therefore, our network contains 244 nodes (Kumcu, 2019), which corresponds to ‘Network 12’ in Table 1. Additionally, on the basis of expert opinions from the Turkish Red Crescent, we generated networks that contain three, five, seven or 10 cities in the region and selected subsets of service providers in these cities. The characteristics of the networks corresponding to each scenario are shown in Table 1. We use Networks 1–6 for determining the parameters of our algorithm and carry out the computational experiments in Networks 7–12.

According to UNHCR (2019), a minimum of 45 square meters per person are required in refugee camps. Camp capacities are determined by expert views based on the size of the available area for the camp and change between 5,000 and 25,000 refugees per camp.

The f_i parameter in the objective function of the PM model captures the trade-off between the setup cost of camps and the lifetime routing costs. It is usually assumed by the professionals in the field that $f_i = 0$ for all $i \in R$ because the lands are provided by the government. In our analysis, we additionally tested $f_i = wq_i$, where q_i is the capacity of camp i and w is equal to the land cost of 45 square meters per refugee, divided by the lifetime routing cost per kilometer. We obtained the land costs from the Turkish Revenue Agency (2019). Considering three visits per week and a lifetime of five to 10 years, the range of values we tested for w therefore changes between 0 and 0.05. We refer to w as the trade-off weight.

The number C_h of service providers in hosting institution $h \in H$ generally does not constitute a restriction in the region. For this reason, we consider loose capacities in the initial set of runs. We also tested tight capacities to show the efficiency of our algorithms by setting all hosting institution capacities equal to 1 in Section 4.5.

In our computational experiments, we assume a fixed value L as the maximum number of camps that can be visited from every host. That is, the hop bound $L_h = L$ for $h \in H$. For a given network, we use the phrase basic settings to refer to setting with hop bound $L \in \{3, 4, 5\}$, the refugee number $Q \in \{50K, 100K, 150K, 200K\}$ and $w \in \{0, 0.005, 0.010, 0.025, 0.050\}$, which

401 yields a total of 60 instances.

402 4.2. Impacts of elementarity assumption on the LP relaxations of the model

403 The strength of the root node relaxation is a critical factor on the solution efficiency of the
404 PM model. The correctness of the model is ensured under different elementarity assumptions
405 made on the paths, however these assumptions greatly affect the LP relaxations. We will therefore
406 investigate the LP relaxations under three different elementarity assumptions. We first consider
407 non-elementary paths. The paths can contain cycles and the pricing problem is therefore a non-
408 elementary HSPP in graphs with negative cost cycles, which is solved using the Bellman-Ford
409 algorithm (Ford, 1956; Bellman, 1958) in $O(|N|^3)$ time. We then consider 2-cycle-free paths and
410 elementary paths. We use our CHSPP algorithm for canceling 2-cycles and all cycles. Addition-
411 ally, we also test the impacts of adding knapsack cuts, as presented in Section 3.9. In total we test
412 these four alternatives using Networks 1–6 with the basic settings, which yields a total of 1,440
413 instances.

414 Table 2 shows the average optimality gaps of PM model LP relaxations under non-elementary,
415 2-cycle-free and elementary path assumptions, and under the elementary path assumption with the
416 addition of knapsack cuts. The results are presented for the L , Q and w values considered in the
417 basic settings in Table 2, parts (a), (b) and (c), respectively. Canceling 2-cycles helps improve the
418 LP relaxation significantly with respect to the non-elementary paths for $L = 3$, but this property
419 does not carry over as L increases. On the other hand, the elementary path assumption reduces the
420 gaps with respect to non-elementary path assumption for all L values considered. The elementary
421 path assumption, combined with the addition of knapsack cuts performs the best among all cases
422 considered. In Table 2(b), the gaps improve across the rows and columns. The LP relaxations
423 improve when there is a larger number of refugees, and the average gap reduces to 0.68% in
424 the best setting. The gaps also decrease in Table 2(c) across rows and columns. The impact of
425 considering fixed costs on the LP relaxations is also significant. The relaxations improve greatly
426 when higher cost weights are considered.

427 Considering the improvements obtained through the elementary path assumption and with the
428 addition of knapsack cuts, we use this setting in our experiments. The cut generation helps improve
429 the root node relaxation, but our preliminary results showed that generating cuts in the rest of the
430 B&B tree can be time-consuming. For this reason, we only generate cuts in the first 10 explored
431 nodes of the B&B tree, and we then stop the cut generation.

432 4.3. Determining the parameter settings of the algorithm

433 The efficiency of the BP&C algorithm also depends on the number of variables pricing out at
434 each iteration. Note that the CHSPP algorithm can be easily customized to output multiple paths
435 with negative reduced costs (if they exist), the details of which we omit for the sake of conciseness.
436 Let *mult* be the maximum number of pricing out variables added for each service provider. In this
437 section, we test the efficiency of the algorithm when *mult* = 1, 5, 10, 50 or 100 for each service
438 provider. Additionally, the impact of the variable fixing as introduced in Section 3.10.1 is tested.

439 To this end, we use Networks 1–6 with the basic settings to test variable fixing (*true* and
440 *false*) and 5 *mult* values (1, 5, 10, 50 and 100), which yields a total of 3,600 instances. Table
441 3 presents the average solution times and the optimality gaps. Variable fixing always helps in

Table 2: Average optimality gaps (%) of LP relaxations of the PM model under non-elementary, 2-cycle-free and elementary path assumptions and under elementary path assumption with the addition of knapsack cuts

(a) Results by changing values of maximum camps in a single tour

L	Non-elementary	2-cycle-free	Elementary	Elementary with cuts
3	5.93	3.94	3.94	3.23
4	6.56	5.42	3.60	3.00
5	8.46	6.17	3.72	3.06

(b) Results by varying number of refugees

Q	Non-elementary	2-cycle-free	Elementary	Elementary with cuts
50	10.26	9.10	9.09	6.87
100	6.44	4.44	1.95	1.90
150	7.01	5.35	3.12	2.93
200	4.21	1.81	0.86	0.68

(c) Results by different tradeoff weights

w	Non-elementary	2-cycle-free	Elementary	Elementary with cuts
0	15.89	10.92	7.07	4.94
0.005	7.93	6.00	4.44	3.86
0.010	5.76	4.53	3.56	3.21
0.025	3.36	2.77	2.31	2.16
0.050	1.97	1.66	1.41	1.32

Table 3: Average solution times and optimality gaps for different number of pricing out variables with and without variable fixing

(a) Average solution times (s)

Var Fixing	<i>mult</i>				
	1	5	10	50	100
<i>false</i>	1531.3	1498.7	1485.7	1460.6	1453.3
<i>true</i>	1444.7	1404.5	1389.4	1375.7	1371.7

(b) Average optimality gaps (%)

Var Fixing	<i>mult</i>				
	1	5	10	50	100
<i>false</i>	0.016%	0.017%	0.017%	0.017%	0.017%
<i>true</i>	0.013%	0.014%	0.015%	0.015%	0.015%

442 accelerating the solution process and in closing the optimality gaps. However, as *mult* increases,
 443 the solution times decrease and optimality gaps increase. This is explained by the instances that
 444 are solved optimally within a very short period of time. Adding a bulk of pricing out variables
 445 helps solve these instances very fast. However, as the B&B tree grows, generating several path
 446 variables becomes a burden due to the increasing sizes of the LPs that are solved. When only those
 447 instances taking more than 360 seconds are taken into account, then the average solution times are
 448 6530.0, 6391.2, 6681.3, 7217.9 and 7635.6 seconds for *mult* = 1, 5, 10, 50 and 100, respectively.
 449 We therefore conclude that setting *mult* = 5 balances both affects. We also turn on the variable
 450 fixing in all nodes of the B&B tree.

451 4.4. Computational results

452 After having fixed the algorithm parameters, we carried out a computational study on Net-
 453 works 7–12 using the basic settings, which amounts to 360 instances. Table 4 shows the results
 454 categorized by network, L , Q and w . In total, 352 instances were solved to optimality within
 455 the time limit with an average of 2119.4 seconds and an average optimality gap of 0.01%. The
 456 optimality gap of the unsolved instances is 0.44% on average, with a minimum of 0.06% and a
 457 maximum of 1.29%. In Table 4(a), Networks 9 and 12 have the longest average solution times
 458 and non-zero average optimality gaps. The reason is that these two networks have the highest
 459 numbers of service hosts (Table 1). Even though Networks 10 and 11 have greater number of
 460 candidate refugee camps, all instances in these two networks could be solved to optimality. This
 461 suggests that the number of service hosts complicates the solution process more than the number
 462 of refugee camps. Table 4(b) reports the results as a function of L . The solution times are higher
 463 for both $L = 3$ and $L = 5$ when compared to $L = 4$, but the reasons are different. When $L = 3$, the
 464 pricing problem is easier to solve, but the algorithm spends more time in the B&B tree, whereas
 465 when $L = 5$, the pricing problem takes longer to solve, but the B&B tree is smaller due to smaller
 466 root node relaxations. The results presented in Table 4(c) and (d) relate to change in the Q and w
 467 parameters. They confirm the robustness of the algorithm for various inputs.

468 In order to rigorously test the computational efficiency of the algorithm under different settings,
 469 we build new problem instances by modifying the camp capacities, the host capacities and the
 470 number of services in the data. For the camp capacities, we consider loose and tight settings. The
 471 loose setting corresponds to the data, as considered in the previous set of runs. In the tight setting,
 472 we halve the capacities. Similarly, we consider a loose and a tight setting for host capacities.
 473 The loose setting corresponds to the data and the tight setting is obtained by assigning a single
 474 service provider for each host. In the data, we have 3 services to be provided to the refugees. In
 475 this section, we test our algorithm for 1, 3 and 5 service types by keeping the number of service
 476 providers the same. In total we have 12 different settings, each of which is applied on Network 11
 477 and for $L_h \in \{1, 2, 3, 4, 5\}$, $Q \in \{50K, 100K, 150K, 200K\}$ and $w \in \{0, 0.005, 0.010, 0.025, 0.050\}$.
 478 This makes a total of 1200 new instances.

479 The results are reported in Table 5. The first four columns are the settings and we report
 480 the number of solved instances, solution times in seconds and the average optimality gaps in
 481 columns 5–7, respectively. The algorithm optimally solved 1057 of the 1200 instances (88.1%).
 482 The average solution time is 4480.7 seconds and the average optimality gap is 0.15% over all the
 483 instances. Camp capacities have a strong impact on the solution efficiency. Instances with tight

Table 4: Number of solved instances, solution times (s) and average optimality gaps (%) for Networks 7–12

(a) Results by networks			
Network	No. solved	Solution time (s)	Gap (%)
7	60	1651.4	0
8	60	1096.4	0
9	56	2823.6	0.013
10	60	1041.8	0
11	60	1951.9	0
12	56	4151.5	0.046

(b) Results by different number of hops			
L	No. solved	Solution time (s)	Gap (%)
3	116	1948.2	0.006
4	120	581.0	0
5	116	3829.1	0.023

(c) Results by different number of refugees			
Q	No. solved	Solution time (s)	Gap (%)
50K	86	2555.0	0.031
100K	90	1241.6	0
150K	90	1316.5	0
200K	86	3364.7	0.008

(d) Results by different tradeoff weights			
w	No. solved	Solution time (s)	Gap (%)
0	72	1215.3	0
0.005	70	2098.0	0.013
0.010	70	2333.8	0.021
0.025	70	2524.9	0.010
0.050	70	2425.2	0.005

Total/Avg.	352	2119.4	0.010
------------	-----	--------	-------

484 capacities are harder to solve than the ones with loose capacities. The total number of solved
485 instances for loose and tight camp capacity settings are 549 and 508, respectively. The average
486 solution time is 3740.1 seconds for loose camp capacities, but it increases to 5215.3 seconds in the
487 tight setting. We have not observed a similar impact for the host capacities. For the loose and tight
488 settings of host capacities, the total number of solved instances are 528 and 529, respectively and
489 the average solution times are 4528.8 and 5525.4 seconds, respectively. The number of services
490 also have a strong negative impact on the solution times, particularly when the number of services
491 increases to 5. The total number of instances solved optimally is 387, 383 and 287 when the
492 number of services equals 1, 3 and 5, respectively. The average solution times are 2720.9, 2769.3
493 and 7949.1 seconds respectively.

494 In summary, we have tested the efficiency of our algorithm on networks with different numbers
495 of camps (18–74) and hosts (12–170), under various settings by changing the refugee numbers
496 (50K–200K), number of services (1–5), camp and host capacities (loose and tight), cost param-
497 eters (0–0.050) and number of hop bounds (1–5). The computational experiments show that our
498 algorithm is robust in solving the problem with various inputs.

Table 5: Number of solved instances, solution times (s) and average optimality gaps (%) for Networks 11 with different camp capacity, host capacity and service settings

Setting number	Camp capacity	Host capacity	No. of services	No. Solved	Solution time (s)	Gap (%)
1	L	L	1	95	3481.3	0.01
2	L	L	3	100	1472.7	0
3	L	L	5	77	6711.3	0.50
4	L	T	1	92	3527.1	0.02
5	L	T	3	100	1900.9	0
6	L	T	5	85	5347.6	0.25
7	T	L	1	100	1925.0	0
8	T	L	3	100	2576.0	0
9	T	L	5	56	11006.4	0.57
10	T	T	1	100	1950.3	0
11	T	T	3	83	5127.6	0.06
12	T	T	5	69	8741.9	0.41
Average				88.1	4480.7	0.15

499 4.5. Discussion

500 In this section, we analyze the results on real-world data, corresponding to Network 12 and
501 for $L_h \in \{1, 2, 3, 4, 5\}$, $Q \in \{50K, 100K, 150K, 200K\}$ and $w \in \{0, 0.005, 0.010, 0.025, 0.050\}$, for
502 a total of 100 instances. Additionally, we solve the same instances with tight host capacities by
503 assigning a single service provider for each host. The detailed results are presented in Tables 10
504 and 11 in the appendix, for the uncapacitated and capacitated instances, respectively. The objective
505 function values for capacitated and uncapacitated scenarios are presented in Figure 2 for varying
506 L values. The costs decrease sharply if more than one camp can be visited in a single tour. This

507 change is not as significant when $L \geq 2$. We have not observed much difference in terms of
 508 objective function value between the capacitated and uncapacitated scenarios, except when $L = 1$.

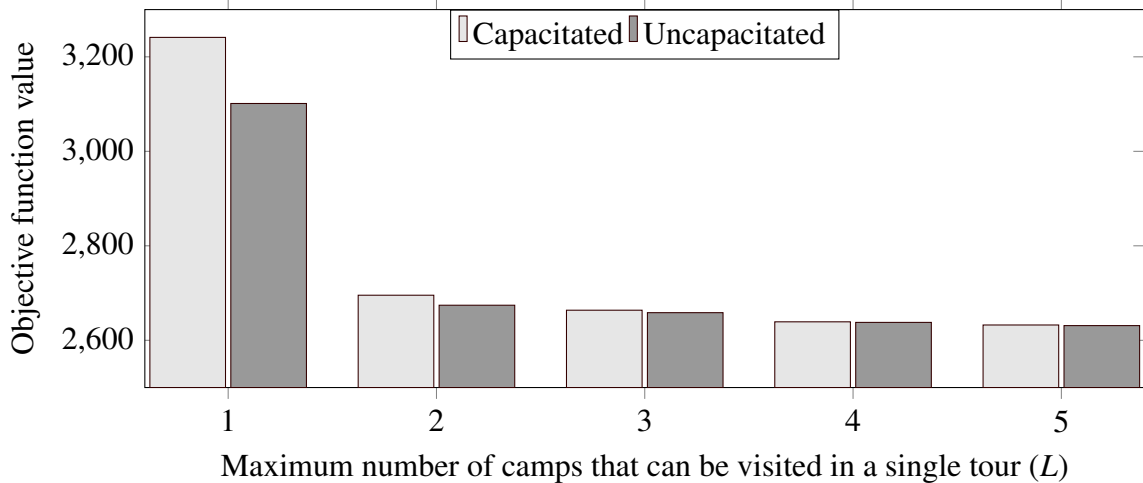


Figure 2: Objective function value change for different L values with capacitated and uncapacitated service hosts

509 Table 6 presents the objective function resulting from changes in L and Q . For $Q = 50K$,
 510 visiting more than two camps in a single tour does not affect the objective function. The marginal
 511 improvement decreases as L increases. Figure 3 plots the average objective value as a function
 512 of Q . The cost increases almost linearly as a function of the number of refugees. This result is
 513 mainly due to the fact that the fixed costs also change linearly with increases in the number of
 514 refugees because of the 45 square meter requirement. The increase in the objective function value
 515 is significant when the trade-off weight w increases (Table 7). This parameter mainly puts more
 516 emphasis on the fixed costs, and the relative weight of routing cost decreases.

Table 6: Objective function values for changing L and Q

L	Q			
	50K	100K	150K	200K
1	1125.23	2414.04	3804.32	5341.24
2	1007.27	2057.22	3232.52	4442.34
3	1007.27	2054.20	3203.15	4379.74
4	1007.27	2049.32	3150.63	4346.82
5	1007.27	2049.32	3134.29	4336.16

517 Table 8 shows the number of camps opened in different settings. For a given L and Q setting,
 518 the number of camps does not change when $w \geq 0.01$ because the cost of opening a camp is
 519 highly dominant in the objective function. For a given L and w setting, if a camp appears in the
 520 optimal solution for locating 50K refugees, it keeps appearing when locating higher numbers of
 521 refugees as well. In all the scenarios considered for Network 12, both for the capacitated and the

Table 7: Objective function values for changing w and Q

w	Q			
	50K	100K	150K	200K
0	108.68	322.10	604.98	969.26
0.005	381.18	824.60	1354.98	1969.26
0.010	638.15	1325.80	2104.98	2969.26
0.025	1388.15	2825.80	4354.98	5969.26
0.050	2638.15	5325.80	8104.98	10969.26

Table 8: Number of camps opened in different instances

w	L	Q - Uncapacitated				Q - Capacitated			
		50K	100K	150K	200K	50K	100K	150K	200K
0	1	4	6	8	10	3	5	8	10
	2	3	6	8	12	3	6	8	10
	3	3	6	10	12	3	6	8	12
	4	3	6	10	12	3	6	10	12
	5	3	6	10	12	3	6	10	12
0.005	1	4	6	8	10	3	5	8	10
	2	3	6	8	12	3	6	8	10
	3	3	6	10	12	3	6	8	12
	4	3	6	10	12	3	6	10	12
	5	3	6	10	12	3	6	10	12
0.010	1	4	6	8	10	4	6	8	10
	2	2	6	8	12	2	6	8	10
	3	2	6	10	12	2	6	8	12
	4	2	6	10	12	2	6	10	12
	5	2	6	10	12	2	6	10	12
0.025	1	4	6	8	10	4	6	8	10
	2	2	6	8	12	2	6	8	10
	3	2	6	10	12	2	6	8	12
	4	2	6	10	12	2	6	10	12
	5	2	6	10	12	2	6	10	12
0.050	1	4	6	8	10	4	6	8	10
	2	2	6	8	12	2	6	8	10
	3	2	6	10	12	2	6	8	12
	4	2	6	10	12	2	6	10	12
	5	2	6	10	12	2	6	10	12

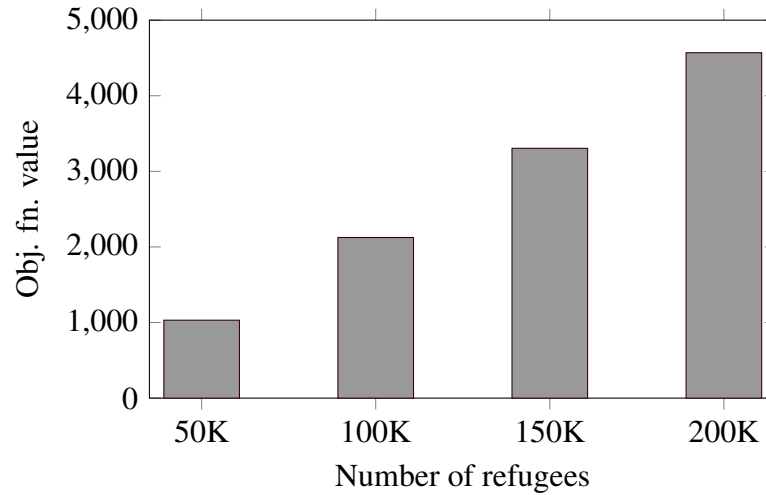


Figure 3: Objective function values for different numbers of refugees

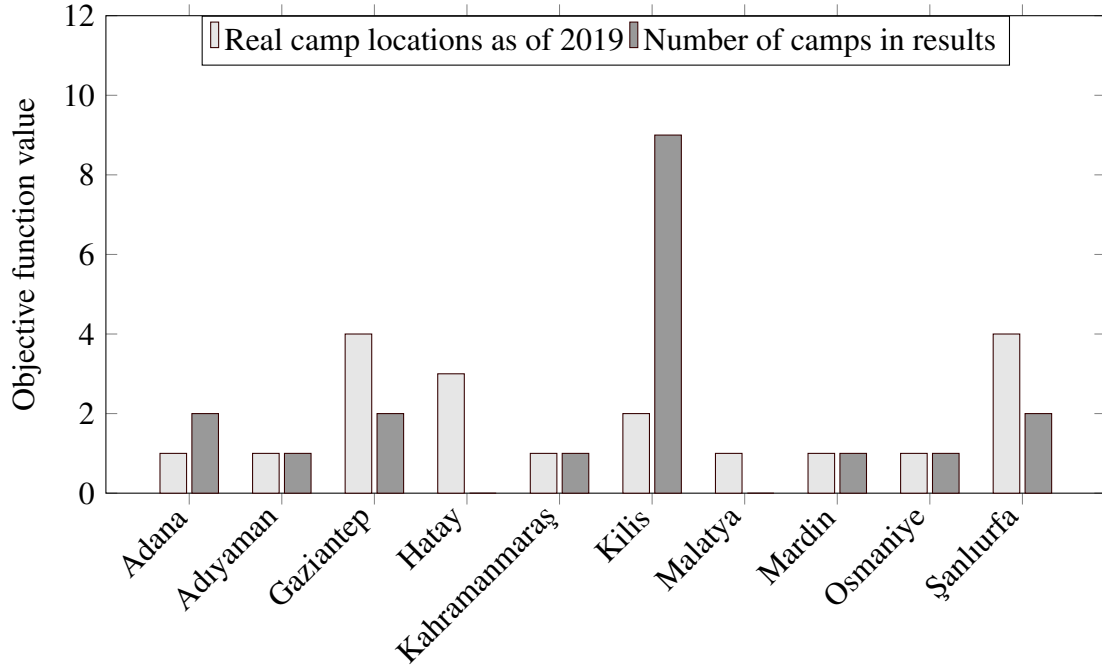
522 uncapacitated versions, the camps consistently appear in the optimal solutions 99.48% of the time
 523 after their initial appearance. A total of 19 camps among the 74 candidate locations appear in
 524 the optimal solutions and the frequency of their appearance is shown in Table 9. The number of
 525 refugee camps in the southeastern part of Turkey changed between 19 and 24 from October 2019
 526 to August 2018 (UNHCR, 2019). The number of refugees in August 2018 was 203,300 in 19
 527 open camps (UNHCR, 2019). The number of camps in each city is compared to the cities in our
 528 solutions in Figure 4. A large difference is observed in Kilis. The camps our algorithm locates in
 529 Kilis are distributed between Gaziantep, Hatay and Şanlıurfa in the real world.

530 5. Conclusions

531 Inspired by a real-world problem, we have defined, modeled and solved the refugee camp
 532 location and public service planning problem. The problem brings a new facet to the location and
 533 routing literature by optimizing the user locations. This feature has the potential to model public
 534 sector service planning when the planning authorities have the possibility of deciding both the
 535 location of the service providers and those of the service beneficiaries. Such applications include
 536 public service planning when a large population evacuates a region and when basic services need
 537 to be provided for the evacuees. We have modeled this problem and developed an efficient exact
 538 decomposition algorithm by exploiting features of the model. An extensive computational study
 539 has shown that our BP&C algorithm can scale up to networks with as many as 244 nodes. Future
 540 research can potentially take into account equity among the cities, the origins of the refugees and
 541 refugee urban locations, among others.

Table 9: Frequency of camps appearing in the optimal solutions

Camp number	City	Frequency (%)
5	Kilis	100
9	Kilis	99
6	Kilis	97.5
1	Kilis	96
69	Kahramanmaraş	95
51	Adana	83
4	Kilis	78.5
2	Kilis	55
11	Kilis	40
3	Kilis	38
62	Adıyaman	30
17	Gaziantep	27.5
53	Adana	25
37	Şanlıurfa	22.5
30	Osmaniye	20.5
10	Kilis	20
36	Şanlıurfa	17.5
68	Mardin	6.5
13	Gaziantep	5



Maximum number of camps that can be visited in a single tour (L)

Figure 4: Objective function value for different number of hops for capacitated and uncapacitated versions

542 Acknowledgment

543 The authors gratefully acknowledge funding provided by the Canadian Natural Sciences and
 544 Engineering Research Council under grant 2015-06189 and by the Scientific and Technological
 545 Research Council of Turkey (TUBITAK) under the grant number 216M380. [Thanks are due to the](#)
 546 [Associate Editor and to the referees for their valuable comments.](#)

547 Appendix: Detailed results

548 We present in this appendix detailed results for instances with uncapacitated (Table 10) and
 549 capacitated (Table 11) service hosts. In both tables, the first four columns are parameters: # is
 550 the instance number, w is the trade-off weight, L is the maximum number of camps that can be
 551 visited on a single tour and Q is the number of refugees. We report the results in the following
 552 eight columns. Root node relaxation is the LP relaxation at the root node after the addition of the
 553 knapsack cover inequalities. Lower bound, upper bound, optimality gap, solution time, number
 554 of variables, number of cuts, and the number of B&B tree nodes are reported in columns 6–12,
 555 respectively.

556 **References**

- 557 Albareda-Sambola, M., 2015. Location-routing and location-arc routing. Springer, Cham. pp. 399–418.
- 558 Altay, N., Green, W.G., 2006. OR/MS research in disaster operations management. *European Journal of Operational*
559 *Research* 175, 475–493.
- 560 Anadolu Agency, 2020. Turkey: 77,000 expats successfully finished quarantines. [Www.aa.com.tr/en/health/turkey-](http://www.aa.com.tr/en/health/turkey-77-000-expats-successfully-finished-quarantines/1863365)
561 [77-000-expats-successfully-finished-quarantines/1863365](http://www.aa.com.tr/en/health/turkey-77-000-expats-successfully-finished-quarantines/1863365), Last accessed Sep 28, 2020.
- 562 Balcik, B., 2017. Site selection and vehicle routing for post-disaster rapid needs assessment. *Transportation Research*
563 *Part E: Logistics and Transportation Review* 101, 30–58.
- 564 Balcik, B., Beamon, B.M., 2008. Facility location in humanitarian relief. *International Journal of Logistics* 11,
565 101–121.
- 566 Balcik, B., Beamon, B.M., Smilowitz, K.R., 2008. Last mile distribution in humanitarian relief. *Journal of Intelligent*
567 *Transportation Systems* 12, 51–63.
- 568 Bellman, R., 1958. On a routing problem. *Quarterly of Applied Mathematics* 16, 87–90.
- 569 Besiou, M., Van Wassenhove, L.N., 2019. Humanitarian operations: A world of opportunity for relevant and impactful
570 research. *Manufacturing & Service Operations Management* doi:10.1287/msom.2019.0799. ahead of print.
- 571 Bettinelli, A., Ceselli, A., Righini, G., 2011. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous
572 vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies* 19, 723–740.
- 573 Campbell, A.M., Vandenbussche, D., Hermann, W., 2008. Routing for relief efforts. *Transportation Science* 42,
574 127–145.
- 575 Çelik, M., Ergun, Ö., Johnson, B., Keskinocak, P., Lorca, Á., Pekgün, P., Swann, J., 2012. Humanitarian logistics, in:
576 *New directions in informatics, optimization, logistics, and production*, pp. 18–49.
- 577 Cherkesly, M., Rancourt, M.È., Smilowitz, K.R., 2019. Community healthcare network in underserved areas: Design,
578 mathematical models, and analysis. *Production and Operations Management* 28, 1716–1734.
- 579 Costa, L., Contardo, C., Desaulniers, G., 2019. Exact branch-price-and-cut algorithms for vehicle routing. *Trans-*
580 *portation Science* 53, 946–985.
- 581 Dahl, G., Gouveia, L., 2004. On the directed hop-constrained shortest path problem. *Operations Research Letters* 32,
582 15–22.
- 583 Desrochers, M., Desrosiers, J., Solomon, M.M., 1992. A new optimization algorithm for the vehicle routing problem
584 with time windows. *Operations Research* 40, 342–354.
- 585 Desrosiers, J., Lübbecke, M.E., 2005. *A Primer in Column Generation*. Springer US, Boston. pp. 1–32.
- 586 Drexl, M., Schneider, M., 2017. A survey of the standard location-routing problem. *Annals of Operations Research*
587 259, 389–414.
- 588 Eisenhandler, O., Tzur, M., 2019a. The humanitarian pickup and distribution problem. *Operations Research* 67,
589 10–32.
- 590 Eisenhandler, O., Tzur, M., 2019b. A segment-based formulation and a matheuristic for the humanitarian pickup and
591 distribution problem. *Transportation Science* 53, 1389–1408.
- 592 Feillet, D., 2010. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR* 8, 407–
593 424.
- 594 Ford, L.R., 1956. *Network flow theory*. Technical Report. Rand Corp, Santa Monica, CA.
- 595 Galindo, G., Batta, R., 2013. Review of recent developments in OR/MS research in disaster operations management.
596 *European Journal of Operational Research* 230, 201–211.
- 597 Huang, M., Smilowitz, K.R., Balcik, B., 2012. Models for relief routing: Equity, efficiency and efficacy. *Transporta-*
598 *tion Research Part E: Logistics and Transportation Review* 48, 2–18.
- 599 Huang, M., Smilowitz, K.R., Balcik, B., 2013. A continuous approximation approach for assessment routing in
600 disaster relief. *Transportation Research Part B: Methodological* 50, 20–41.
- 601 Irnich, S., Desaulniers, G., 2005. *Shortest Path Problems with Resource Constraints*. Springer US, Boston, MA. pp.
602 33–65.
- 603 Kara, B.Y., Savaşer, S., 2017. Humanitarian logistics, in: *Leading Developments from INFORMS Communities*, pp.
604 263–303.
- 605 Karsu, O., Kara, B.Y., Selvi, B., 2019. The refugee camp management: a general framework and a unifying decision-
606 making model. *Journal of Humanitarian Logistics and Supply Chain Management* 9, 131–150.

607 Kumcu, G.C., 2019. Location-Location Routing Problem and Its Application on Refugee Camps. Master's thesis.
608 Bilkent University.

609 Lübbecke, M.E., Desrosiers, J., 2005. Selected topics in column generation. *Operations Research* 53, 1007–1023.

610 Mills, A.F., Argon, N.T., Ziya, S., 2018. Dynamic distribution of patients to medical facilities in the aftermath of a
611 disaster. *Operations Research* 66, 716–732.

612 Murray-Tuite, P., Wolshon, B., 2013. Evacuation transportation modeling: An overview of research, development,
613 and practice. *Transportation Research Part C: Emerging Technologies* 27, 25–45.

614 Oruc, B.E., Kara, B.Y., 2018. Post-disaster assessment routing problem. *Transportation research part B: methodological*
615 *116*, 76–102.

616 Ozbaygin, G., Karasan, O.E., Savelsbergh, M., Yaman, H., 2017. A branch-and-price algorithm for the vehicle routing
617 problem with roaming delivery locations. *Transportation Research Part B: Methodological* 100, 115–137.

618 Ozbaygin, G., Savelsbergh, M., 2019. An iterative re-optimization framework for the dynamic vehicle routing problem
619 with roaming delivery locations. *Transportation Research Part B: Methodological* 128, 207–235.

620 Paul, J.A., Wang, X., 2019. Robust location-allocation network design for earthquake preparedness. *Transportation*
621 *Research Part B: Methodological* 119, 139–155.

622 Prodhon, C., Prins, C., 2014. A survey of recent research on location-routing problems. *European Journal of Opera-*
623 *tional Research* 238, 1–17.

624 Rothenbächer, A.K., Drexl, M., Irnich, S., 2018. Branch-and-price-and-cut for the truck-and-trailer routing problem
625 with time windows. *Transportation Science* 52, 1174–1190. doi:10.1287/trsc.2017.0765.

626 Sherali, H.D., Carter, T.B., Hobeika, A.G., 1991. A location-allocation model and algorithm for evacuation planning
627 under hurricane/flood conditions. *Transportation Research Part B: Methodological* 25, 439–452.

628 Sheu, J.B., 2014. Post-disaster relief-service centralized logistics distribution with survivor resilience maximization.
629 *Transportation Research Part B: Methodological* 68, 288–314.

630 Turkish Revenue Agency, 2019. Square meter costs of minimal unit lands in 2014. intvd.gib.gov.tr/2014_Emlak_Arsa,
631 Last accessed: Nov 10, 2019.

632 UN/DESA, 2019. World economic situation and prospects 2019. [www.un.org/development/desa/dpad/wp-](http://www.un.org/development/desa/dpad/wp-content/uploads/sites/45/WESP2019_BOOK-web.pdf)
633 [content/uploads/sites/45/WESP2019_BOOK-web.pdf](http://www.un.org/development/desa/dpad/wp-content/uploads/sites/45/WESP2019_BOOK-web.pdf), Last accessed May 8, 2019.

634 UNHCR, 2019. Camp planning standards. emergency.unhcr.org/entry/45582, Last accessed: Nov 16, 2019.

635 UNHCR, 2019. Operations portal refugee situations. data2.unhcr.org/en/situations, Last accessed Dec 3, 2019.

636 UNHCR, 2019. Population statistics database. popstats.unhcr.org/en/, Last accessed May 8, 2019.

637 UNHCR, 2019. Statistical yearbooks. www.unhcr.org/figures-at-a-glance.html, Last accessed May 8, 2019.

638 UNHCR, 2019. UNHCR in Turkey www.unhcr.org/tr/en/unhcr-in-turkey, Last accessed Nov 11, 2019.

639 UNHCR, 2019. UNHCR Turkey-Syrian refugee camps and provincial breakdown of Syrian refugees registered in
640 south east Turkey - september 2018. data2.unhcr.org/en/documents/details/65578, Last accessed Dec 3, 2019.

641 UNICEF, 2011. Guidelines for child friendly spaces in emergencies.
642 www.unicef.org/protection/Child_Friendly_Spaces_Guidelines_for_Field_Testing.pdf, Last accessed Feb 15,
643 2018.

644 Van Wassenhove, L.N., 2006. Humanitarian aid logistics: supply chain management in high gear. *Journal of the*
645 *Operational Research Society* 57, 475–489.

646 Wang, Z., Sheu, J.B., 2019. Vehicle routing problem with drones. *Transportation Research Part B: Methodological*
647 *122*, 350–364.

648 Wolsey, L.A., 1975. Faces for a linear inequality in 0–1 variables. *Mathematical Programming* 8, 165–178.

649 Yi, W., Özdamar, L., 2007. A dynamic logistics coordination model for evacuation and support in disaster response
650 activities. *European Journal of Operational Research* 179, 1177–1193.

651 Özdamar, L., Ertem, M.A., 2015. Models, solutions and enabling technologies in humanitarian logistics. *European*
652 *Journal of Operational Research* 244, 55–65.

Table 10: Results for Network 12 with uncapacitated service hosts

#	w	L	Q	Root node relaxation	Lower bound	Upper bound	Optimality gap (%)	Solution Time (s)	No. of Variables	No. of Cuts	No. of B&B nodes
1	0	1	50K	166.71	166.71	166.71	0	0.8	466	1	0
2	0	1	100K	543.54	543.54	543.54	0	0.9	466	1	0
3	0	1	150K	1028.93	1028.93	1028.93	0	0.6	466	0	0
4	0	1	200K	1660.21	1665.85	1665.85	0	3.0	466	6	5
5	0	2	50K	78.34	83.66	83.66	0	100.3	3408	7	5
6	0	2	100K	253.67	255.1	255.1	0	81.4	3806	4	3
7	0	2	150K	510.79	514.08	514.08	0	116.0	4027	2	13
8	0	2	200K	814.19	820.54	820.54	0	100.9	3463	5	7
9	0	3	50K	78.34	83.66	83.66	0	492.1	8879	5	7
10	0	3	100K	238.04	252.77	252.77	0	577.7	9786	4	9
11	0	3	150K	473.86	495.39	495.39	0	1074.5	11536	9	27
12	0	3	200K	775.14	778.5	778.5	0	532.4	10071	4	15
13	0	4	50K	78.34	83.66	83.66	0	1652.8	12812	7	7
14	0	4	100K	236.73	249.32	249.32	0	1617.8	10977	7	5
15	0	4	150K	449.47	449.47	449.47	0	794.0	9591	1	0
16	0	4	200K	744.38	745.57	745.57	0	1025.7	10957	1	5
17	0	5	50K	78.34	83.66	83.66	0	12759.7	10217	8	9
18	0	5	100K	236.46	249.32	249.32	0	9830.8	10517	7	5
19	0	5	150K	432.94	432.94	432.94	0	2324.8	15231	0	0
20	0	5	200K	733.55	734.92	734.92	0	13551.2	19013	5	7
21	0.005	1	50K	416.71	416.71	416.71	0	1.0	466	1	0
22	0.005	1	100K	1043.54	1043.54	1043.54	0	0.9	466	1	0
23	0.005	1	150K	1778.93	1778.93	1778.93	0	0.7	466	0	0
24	0.005	1	200K	2660.21	2665.85	2665.85	0	3.2	466	6	5
25	0.005	2	50K	328.34	358.66	358.66	0	159.9	3351	11	11
26	0.005	2	100K	753.67	755.1	755.1	0	87.2	3385	4	3
27	0.005	2	150K	1260.79	1264.08	1264.08	0	101.4	2964	2	13
28	0.005	2	200K	1814.19	1820.54	1820.54	0	114.0	4135	5	7
29	0.005	3	50K	328.34	358.66	358.66	0	848.2	11666	12	15
30	0.005	3	100K	738.04	752.77	752.77	0	575.4	10017	7	9
31	0.005	3	150K	1223.86	1245.39	1245.39	0	1010.2	11325	11	27
32	0.005	3	200K	1775.14	1778.5	1778.5	0	527.0	9707	4	15
33	0.005	4	50K	328.34	358.66	358.66	0	2621.3	16117	12	15
34	0.005	4	100K	736.73	749.32	749.32	0	1534.0	12801	7	5
35	0.005	4	150K	1199.47	1199.47	1199.47	0	710.1	10198	1	0
36	0.005	4	200K	1744.38	1745.57	1745.57	0	965.2	11655	0	5
37	0.005	5	50K	328.34	356.66	358.66	0.56	21712.4	13919	11	11
38	0.005	5	100K	736.46	749.32	749.32	0	8627.5	10994	7	5
39	0.005	5	150K	1182.94	1182.94	1182.94	0	3053.2	12731	0	0
40	0.005	5	200K	1733.87	1734.92	1734.92	0	6106.5	17312	3	3
41	0.010	1	50K	666.71	666.71	666.71	0	0.8	466	1	0
42	0.010	1	100K	1543.54	1543.54	1543.54	0	0.9	466	1	0
43	0.010	1	150K	2528.93	2528.93	2528.93	0	0.5	466	0	0
44	0.010	1	200K	3660.21	3665.85	3665.85	0	3.0	466	6	5
45	0.010	2	50K	578.34	614.68	614.68	0	281.8	3517	11	17
46	0.010	2	100K	1253.67	1255.1	1255.1	0	83.0	2929	4	3
47	0.010	2	150K	2010.79	2014.08	2014.08	0	103.2	2885	2	13
48	0.010	2	200K	2814.19	2820.54	2820.54	0	159.8	3553	5	7
49	0.010	3	50K	578.34	614.68	614.68	0	892.0	10335	12	15
50	0.010	3	100K	1238.04	1252.77	1252.77	0	649.6	9779	7	9
51	0.010	3	150K	1973.86	1995.39	1995.39	0	1155.2	12733	11	27
52	0.010	3	200K	2775.14	2778.5	2778.5	0	554.1	9779	4	13
53	0.010	4	50K	578.34	614.68	614.68	0	2846.1	14383	12	15
54	0.010	4	100K	1236.73	1249.32	1249.32	0	1647.3	10908	7	5
55	0.010	4	150K	1949.47	1949.47	1949.47	0	692.3	10674	1	0
56	0.010	4	200K	2744.38	2745.57	2745.57	0	1153.4	10220	0	5
57	0.010	5	50K	578.34	606.77	614.68	1.29	22178.6	16441	12	15
58	0.010	5	100K	1236.46	1249.32	1249.32	0	9231.1	10520	7	5
59	0.010	5	150K	1932.94	1932.94	1932.94	0	2531.7	14187	0	0
60	0.010	5	200K	2733.87	2734.92	2734.92	0	5612.2	15822	3	3
61	0.025	1	50K	1416.71	1416.71	1416.71	0	0.9	466	1	0
62	0.025	1	100K	3043.54	3043.54	3043.54	0	1.1	466	1	0
63	0.025	1	150K	4778.93	4778.93	4778.93	0	0.6	466	0	0
64	0.025	1	200K	6660.21	6665.85	6665.85	0	3.3	466	6	5
65	0.025	2	50K	1328.34	1364.68	1364.68	0	237.7	3844	12	17
66	0.025	2	100K	2753.67	2755.1	2755.1	0	83.9	3882	4	3
67	0.025	2	150K	4260.79	4264.08	4264.08	0	128.2	3052	2	13
68	0.025	2	200K	5814.19	5820.54	5820.54	0	111.0	4291	5	7
69	0.025	3	50K	1328.34	1364.68	1364.68	0	818.8	10062	12	15
70	0.025	3	100K	2738.04	2752.77	2752.77	0	660.5	12299	7	9
71	0.025	3	150K	4223.86	4245.39	4245.39	0	1049.6	12852	10	25
72	0.025	3	200K	5775.14	5778.5	5778.5	0	550.2	11079	4	13
73	0.025	4	50K	1328.34	1364.68	1364.68	0	3172.3	15150	11	17
74	0.025	4	100K	2736.73	2749.32	2749.32	0	1723.8	14154	7	5
75	0.025	4	150K	4199.47	4199.47	4199.47	0	761.7	8398	1	0
76	0.025	4	200K	5744.38	5745.57	5745.57	0	1288.5	13276	2	5
77	0.025	5	50K	1328.34	1356.66	1364.68	0.59	21664.1	14912	12	14
78	0.025	5	100K	2736.46	2749.32	2749.32	0	9035.9	6603	7	5
79	0.025	5	150K	4182.94	4182.94	4182.94	0	2577.7	13474	0	0
80	0.025	5	200K	5733.55	5734.92	5734.92	0	12290.0	15235	5	7
81	0.050	1	50K	2666.71	2666.71	2666.71	0	1.0	466	1	0
82	0.050	1	100K	5543.54	5543.54	5543.54	0	1.0	466	1	0
83	0.050	1	150K	8528.93	8528.93	8528.93	0	0.7	466	0	0
84	0.050	1	200K	11660.21	11665.85	11665.85	0	3.1	466	6	5
85	0.050	2	50K	2578.34	2614.68	2614.68	0	196.2	4061	11	17
86	0.050	2	100K	5253.67	5255.1	5255.1	0	80.1	3271	4	3
87	0.050	2	150K	8010.79	8014.08	8014.08	0	124.3	3133	2	13
88	0.050	2	200K	10814.19	10820.54	10820.54	0	115.2	3922	5	7
89	0.050	3	50K	2578.34	2614.68	2614.68	0	923.4	11227	12	15
90	0.050	3	100K	5238.04	5252.77	5252.77	0	596.4	10066	7	9
91	0.050	3	150K	7973.86	7995.39	7995.39	0	911.2	10687	10	25
92	0.050	3	200K	10775.14	10778.5	10778.5	0	527.5	9711	4	13
93	0.050	4	50K	2578.34	2614.68	2614.68	0	2683.1	14548	12	15
94	0.050	4	100K	5236.73	5249.32	5249.32	0	1480.5	8833	7	5
95	0.050	4	150K	7949.47	7949.47	7949.47	0	684.9	8279	1	0
96	0.050	4	200K	10744.38	10745.57	10745.57	0	1285.7	11694	2	5
97	0.050	5	50K	2578.34	2606.52	2614.68	0.31	23124.2	17050	12	12
98	0.050	5	100K	5236.46	5249.32	5249.32	0	8567.8	8340	7	5
99	0.050	5	150K	7932.94	7932.94	7932.94	0	2705.9	14028	0	0
100	0.050	5	200K	10733.87	10734.92	10734.92	0	6338.2	16189	3	3

Table 11: Results for Network 12 with capacitated service hosts

#	w	L	Q	Root node relaxation	Lower bound	Upper bound	Optimality gap (%)	Solution Time (s)	No. of Variables	No. of Cuts	No. of B&B nodes
1	0	1	50K	220.05	250.84	250.84	0	2.4	466	7	11
2	0	1	100K	618.54	657.39	657.39	0	2.4	466	7	11
3	0	1	150K	1179.71	1179.71	1179.71	0	0.5	466	1	0
4	0	1	200K	1816.63	1816.63	1816.63	0	0.4	466	0	0
5	0	2	50K	78.34	83.66	83.66	0	81.8	2795	5	5
6	0	2	100K	259.08	259.33	259.33	0	64.1	3260	4	5
7	0	2	150K	534.88	550.96	550.96	0	98.0	3078	2	13
8	0	2	200K	838.30	864.13	864.13	0	221.7	4779	6	37
9	0	3	50K	78.34	83.66	83.66	0	521.7	8986	7	7
10	0	3	100K	238.86	255.63	255.63	0	690.9	9123	7	9
11	0	3	150K	476.34	510.91	510.91	0	1681.4	12385	8	49
12	0	3	200K	777.62	780.98	780.98	0	778.7	12341	7	9
13	0	4	50K	78.34	83.66	83.66	0	1767.2	10183	7	7
14	0	4	100K	237.40	249.32	249.32	0	1393.8	9992	3	3
15	0	4	150K	451.79	451.79	451.79	0	802.5	9990	1	0
16	0	4	200K	746.86	748.06	748.06	0	1352.0	13041	1	5
17	0	5	50K	78.34	83.66	83.66	0	20983.4	12824	9	9
18	0	5	100K	237.00	249.32	249.32	0	9202.4	11723	7	5
19	0	5	150K	435.64	435.64	435.64	0	2975.0	16823	0	0
20	0	5	200K	736.38	737.4	737.4	0	8964.3	17148	2	5
21	0.005	1	50K	470.05	525.84	525.84	0	3.8	466	7	23
22	0.005	1	100K	1118.54	1182.39	1182.39	0	2.9	466	7	15
23	0.005	1	150K	1929.71	1929.71	1929.71	0	0.5	466	1	0
24	0.005	1	200K	2816.63	2816.63	2816.63	0	0.4	466	0	0
25	0.005	2	50K	328.34	358.66	358.66	0	104.0	3536	8	11
26	0.005	2	100K	759.08	759.33	759.33	0	73.7	3779	4	5
27	0.005	2	150K	1284.88	1300.96	1300.96	0	87.0	3298	2	13
28	0.005	2	200K	1838.30	1864.13	1864.13	0	228.2	4856	5	35
29	0.005	3	50K	328.34	358.66	358.66	0	907.6	9731	12	13
30	0.005	3	100K	738.86	755.63	755.63	0	744.2	9644	7	9
31	0.005	3	150K	1226.34	1260.91	1260.91	0	1677.0	13952	8	49
32	0.005	3	200K	1777.62	1780.98	1780.98	0	790.7	10982	4	13
33	0.005	4	50K	328.34	358.66	358.66	0	3308.0	13016	12	15
34	0.005	4	100K	737.40	749.32	749.32	0	1198.2	10411	3	3
35	0.005	4	150K	1201.79	1201.79	1201.79	0	841.3	9469	1	0
36	0.005	4	200K	1746.86	1748.06	1748.06	0	1441.8	13115	2	5
37	0.005	5	50K	328.34	356.66	358.66	0.56	22943.7	13245	12	13
38	0.005	5	100K	737.00	749.32	749.32	0	11266.0	12195	7	5
39	0.005	5	150K	1185.64	1185.64	1185.64	0	3323.3	17152	0	0
40	0.005	5	200K	1736.10	1737.4	1737.4	0	14539.2	20424	5	7
41	0.010	1	50K	720.05	797.37	797.37	0	4.5	466	7	29
42	0.010	1	100K	1618.54	1694.31	1694.31	0	3.8	466	7	23
43	0.010	1	150K	2679.71	2679.71	2679.71	0	0.5	466	1	0
44	0.010	1	200K	3816.63	3816.63	3816.63	0	0.3	466	0	0
45	0.010	2	50K	578.34	614.68	614.68	0	153.3	3691	9	17
46	0.010	2	100K	1259.08	1259.33	1259.33	0	69.8	3580	4	5
47	0.010	2	150K	2034.88	2050.96	2050.96	0	96.3	3339	2	13
48	0.010	2	200K	2838.30	2864.13	2864.13	0	240.6	5589	5	31
49	0.010	3	50K	578.34	614.68	614.68	0	998.3	10250	12	15
50	0.010	3	100K	1238.86	1255.63	1255.63	0	717.9	9021	7	9
51	0.010	3	150K	1976.34	2010.91	2010.91	0	1646.8	12602	8	49
52	0.010	3	200K	2777.62	2780.98	2780.98	0	812.7	13109	5	13
53	0.010	4	50K	578.34	614.68	614.68	0	3222.1	14333	12	15
54	0.010	4	100K	1237.40	1249.32	1249.32	0	1258.9	10314	3	3
55	0.010	4	150K	1951.79	1951.79	1951.79	0	956.1	10120	1	0
56	0.010	4	200K	2746.86	2748.06	2748.06	0	1311.1	13230	0	5
57	0.010	5	50K	578.34	603.49	614.68	1.82	21932.3	12352	12	9
58	0.010	5	100K	1237.00	1249.32	1249.32	0	8908.8	11013	7	5
59	0.010	5	150K	1935.64	1935.64	1935.64	0	3514.2	16504	0	0
60	0.010	5	200K	2736.10	2737.4	2737.4	0	16691.3	20890	5	7
61	0.025	1	50K	1470.05	1547.37	1547.37	0	4.5	466	7	29
62	0.025	1	100K	3118.54	3194.31	3194.31	0	3.9	466	7	23
63	0.025	1	150K	4929.71	4929.71	4929.71	0	0.5	466	1	0
64	0.025	1	200K	6816.63	6816.63	6816.63	0	0.4	466	0	0
65	0.025	2	50K	1328.34	1364.68	1364.68	0	153.9	4412	9	17
66	0.025	2	100K	2759.08	2759.33	2759.33	0	71.7	3947	4	5
67	0.025	2	150K	4284.88	4300.96	4300.96	0	97.8	3386	2	13
68	0.025	2	200K	5838.30	5864.13	5864.13	0	182.7	4642	5	31
69	0.025	3	50K	1328.34	1364.68	1364.68	0	957.4	9854	11	15
70	0.025	3	100K	2738.86	2755.63	2755.63	0	641.7	7896	4	9
71	0.025	3	150K	4226.34	4260.91	4260.91	0	1773.6	11977	8	49
72	0.025	3	200K	5777.62	5780.98	5780.98	0	709.4	10307	5	11
73	0.025	4	50K	1328.34	1364.68	1364.68	0	2993.6	12906	12	15
74	0.025	4	100K	2737.40	2749.32	2749.32	0	1246.9	10074	3	3
75	0.025	4	150K	4201.79	4201.79	4201.79	0	840.2	9247	1	0
76	0.025	4	200K	5746.86	5748.06	5748.06	0	1293.7	11660	0	5
77	0.025	5	50K	1328.34	1356.52	1364.68	0.60	21996.3	11388	12	11
78	0.025	5	100K	2737.00	2749.32	2749.32	0	10154.6	13875	7	5
79	0.025	5	150K	4185.64	4185.64	4185.64	0	2985.7	16406	0	0
80	0.025	5	200K	5736.38	5737.4	5737.4	0	9311.4	20653	2	5
81	0.050	1	50K	2720.05	2797.37	2797.37	0	3.8	466	7	29
82	0.050	1	100K	5618.54	5694.31	5694.31	0	3.9	466	7	23
83	0.050	1	150K	8679.71	8679.71	8679.71	0	0.5	466	1	0
84	0.050	1	200K	11816.63	11816.63	11816.63	0	0.4	466	0	0
85	0.050	2	50K	2578.34	2614.68	2614.68	0	139.8	3853	9	15
86	0.050	2	100K	5259.08	5259.33	5259.33	0	64.4	3636	4	5
87	0.050	2	150K	8034.88	8050.96	8050.96	0	96.3	3402	2	13
88	0.050	2	200K	10838.30	10864.13	10864.13	0	222.3	5021	6	37
89	0.050	3	50K	2578.34	2614.68	2614.68	0	1071.9	11356	12	15
90	0.050	3	100K	5238.86	5255.63	5255.63	0	730.0	9896	7	9
91	0.050	3	150K	7976.34	8010.91	8010.91	0	1950.5	14271	8	49
92	0.050	3	200K	10777.62	10780.98	10780.98	0	718.8	13681	6	9
93	0.050	4	50K	2578.34	2614.68	2614.68	0	3296.3	12670	12	15
94	0.050	4	100K	5237.40	5249.32	5249.32	0	1222.1	10458	3	3
95	0.050	4	150K	7951.79	7951.79	7951.79	0	850.4	10518	1	0
96	0.050	4	200K	10746.86	10748.06	10748.06	0	1497.3	12848	1	5
97	0.050	5	50K	2578.34	2597.85	2614.68	0.64	21963.7	11654	11	9
98	0.050	5	100K	5237.00	5249.32	5249.32	0	10159.3	8666	7	5
99	0.050	5	150K	7935.64	7935.64	7935.64	0	2939.1	14085	0	0
100	0.050	5	200K	10736.10	10737.4	10737.4	0	12925.6	19739	5	7