

# A Branch-Cut-and-Price Approach to the Bus Evacuation Problem with Integrated Collection Point and Shelter Decisions\*

Marc Goerigk, Bob Grün, and Philipp Heßler

Fachbereich Mathematik, Technische Universität Kaiserslautern,  
Germany

June 4, 2013

## Abstract

We consider the problem of evacuating a region with the help of buses. For a given set of possible collection points where evacuees gather, and possible shelter locations where evacuees are brought to, we need to determine both collection points and shelters we would like to use, and bus routes that evacuate the region in minimum time.

We model this integrated problem using an integer linear program, and present a branch-cut-and-price algorithm that generates bus tours in its pricing step. In computational experiments we show that our approach is able to solve instances of realistic size in sufficient time for practical application, and considerably outperforms the usage of a generic ILP solver.

## 1 Introduction

Operations research methods are able to play a vital role in mitigation, preparedness, response, and recovery in natural and man-made disasters. For a general survey on the topic, we refer to [AGI06].

One such aspect is to schedule the evacuation of a specified region with the help of public transport infrastructure, as buses (see [Bis11, GG12, GGH13]). The problem they consider is a vehicle routing problem, in which evacuees need to be transferred from a set of collection points within the endangered area, to a set of shelter points in a safe area, minimizing the time needed for the last person to be brought to safety.

However, this approach assumes that the decision where to pick up evacuees, and where to bring them, has already been made. This corresponds to a two-stage sequential process, in which the travel times can only be estimated during the location planning stage.

---

\*Partially supported by the Federal Ministry of Education and Research Germany, grant DSS.Evac.Logistic, FKZ 13N12229.

In the field of location-routing (see [NS07] for a survey), one integrates both steps into a single problem, which becomes typically more computationally challenging to solve than the sequential approach, but will also result in solutions with superior objective value. In this work we follow this line of research and present an integrated location-routing model for the bus evacuation problem. Following the four key aspects of classification proposed in [NS07], we develop a model with the following properties:

- (a) *Hierarchical structure.* No routes connect facilities (i.e., collection points) with each other. Transport is only between collection points and shelters.
- (b) *Deterministic input data.* Considering the increased complexity of the proposed model, we will not include data uncertainty.
- (c) *Static planning period.* The endangered area needs to be evacuated only once; multiple planning horizons are not necessary.
- (d) *Exact solution method.* Though most papers propose heuristic solution approaches, we consider a branch-cut-and-price scheme that solves the proposed model exactly.

Another specialty of the problem we consider is that evacuees are present at every possible gathering point, and evacuees at gathering points which are not opened need to be allocated to an opened point that is closeby; i.e., apart from the routing aspect to pick these evacuees up, there is also an allocation problem present as part of the location problem.

A location-routing problem with application to evacuation planning has been considered in [CCB01]. They propose a heuristic to determine the location of distribution centers for medical supplies, to allocate a vehicle fleet to supply centers, and to schedule deliveries with stochastically processed demands. A branch-cut-and-price method for a location-routing problem has been proposed in [BBP<sup>+</sup>11]. There, the set of customers is fixed, and a subset of potential facilities can be opened. A fleet of unlimited size and fixed vehicle capacity needs to be routed such that every customer is visited exactly once, minimizing the sum of fixed costs for opening facilities, and routing costs.

**Contributions and outlook.** We introduce the integrated location-routing model for the bus evacuation problem in Section 2. In the following Section 3 and 4 we develop a column generation approach to solve a relaxed path-formulation of the problem, and a branch-and-bound scheme to solve the corresponding integer problem formulation. Furthermore, we develop cuts on the location variables that can be used to strengthen the formulation during the branching process. In computational experiments presented in Section 5, we show that our approach can solve the proposed model in sufficiently small time for real-world applications, and significantly faster than a commercial integer programming solver is able to. IP gaps are reduced from 41.71% to 2.19% on average over all considered instances. Comparing the resulting evacuation time to a sequential approach, we can furthermore evaluate the gain by using an integrated model.

## 2 Problem Description

In the following, we describe in detail the problem we consider. We assume that an emergency situation such as a bomb removal requires the evacuation of a

densely populated region. Such an evacuation will cause both individual traffic to leave the region, as well as organized bus transfers. In this work we focus on the latter.

There are three main decisions that need to be made:

1. Where do we open shelters to accommodate evacuees?
2. Where do we locate collection points for pickup within the affected region?
3. How do we schedule bus transfers?

We assume that there exist discrete sets of possible shelter and collection points; for the former, one would typically use gymnasiums, and for the latter easily recognizable landmarks such as bus stops, schools or pharmacies. We have a fixed number of buses available, which we assume to be equal in terms of capacity and speed. At the beginning of the evacuation, they are standing at a depot. Figure 1 schematizes the considered situation: There are four possible collection points  $\mathcal{S} = \{s_1, \dots, s_4\}$ , and four possible shelter locations  $\mathcal{T} = \{t_1, \dots, t_4\}$ . In the following, we may identify collection point  $s_i$  with  $i$ , or shelter  $t_j$  with  $j$  for short, and use the notation  $[N] = \{1, \dots, N\}$ ; i.e., we will write  $[S]$  for a set of  $S$  collection points, and  $[T]$  for a set of  $T$  shelter locations.

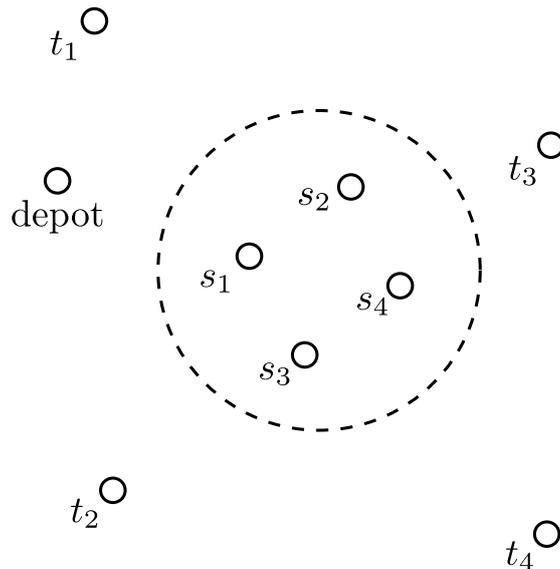


Figure 1: Example problem instance.

Every shelter  $j \in [T]$  has a maximum capacity  $u_j$  of evacuees it can accommodate. Furthermore, every collection point  $i \in [S]$  is assigned a number of evacuees  $l_i$  who live close to that location, and a capacity  $L_i$  that regulates how many people should use that location at most. Due to cost restrictions, and to facilitate the evacuation logistics, the planners decide on the maximum number  $N^{\text{shelter}}$  of shelters that will be opened, and the number  $N^{\text{coll}}$  of collection points.

If we decide not to use a location  $i \in [S]$ , we need to assign these evacuees to a different collection point  $k \in [S]$ , respecting the capacity  $L_k$ . As evacuees

should not need to walk long distances, we assume that we are given a matrix  $(t_{ik}^{\text{walk}})_{i,k \in [S]}$  representing the time needed to go from  $i$  to  $k$ , and a bound  $t^{\text{maxwalk}}$  on the maximal walking time.

Once shelter decisions, collection point decisions, and evacuee assignments are made, we can schedule the bus transport. The time needed to travel from collection point  $i \in [S]$  to shelter  $j \in [T]$  is given by  $d_{ij}$ , and from the depot to collection point  $i \in [S]$  by  $d_i^{\text{start}}$ . A bus route is given by a sequence of collection points and shelters; e.g., the route  $(s_1, t_2, s_2, t_1)$  means that a bus first travels from the depot to collection point  $s_1$ , picks up a bus load of evacuees there, then travels to shelter  $t_2$ , drops off these evacuees, and continues to travel to  $s_2$ . Then it brings a bus load of evacuees from  $s_2$  to  $t_1$ .

Such a bus scheduling problem with fixed location and assignment decisions has already been considered in [Bis11] and [GGH13], and is known as the Bus Evacuation Problem (BEP). It is known to be NP-complete [GG12] and computationally challenging. Integrating the mentioned decisions (which we will call Integrated Bus Evacuation Problem, or IBEP) brings the evacuation model closer to real-world applicability, but also increases the computational complexity (note that IBEP is also NP-complete).

**Example 1** *We reconsider the instance from Figure 1, with the following data:*

$$\begin{aligned}
l &= (3, 3, 3, 3) \\
L &= (6, 6, 6, 6) \\
u &= (9, 8, 3, 4) \\
t^{\text{walk}} &= \begin{pmatrix} 0 & 3 & 2 & 4 \\ 3 & 0 & 7 & 2 \\ 2 & 7 & 0 & 3 \\ 4 & 2 & 3 & 0 \end{pmatrix} \\
d &= \begin{pmatrix} 7 & 7 & 10 & 10 \\ 7 & 10 & 5 & 7 \\ 5 & 3 & 6 & 4 \\ 10 & 11 & 7 & 8 \end{pmatrix} \\
d^{\text{start}} &= (2, 3, 3, 4) \\
B &= 2 \\
N^{\text{coll}} &= 2 \\
N^{\text{shelter}} &= 2 \\
t^{\text{maxwalk}} &= 5
\end{aligned}$$

*An optimal solution is to open shelters  $t_1$  and  $t_2$ , and collection points  $s_1$  and  $s_3$ . We assign the evacuees from  $s_2$  to  $s_1$ , and from  $s_4$  to  $s_3$ . The bus routes we use are*

$$\text{Bus 1: } (1, 1) (1, 2) (3, 2) (3, 2) (3, 2) (1, 1)$$

$$\text{Bus 2: } (1, 2) (3, 2) (3, 2) (3, 2) (1, 1) (1, 1)$$

*with time 55 for both buses.*

We now model the IBEP as an integer linear program. We introduce variables  $y_i^S \in \mathbb{B}$  to determine if collection point  $i \in [S]$  is used, and variables  $y_j^T \in \mathbb{B}$  for

the shelters  $j \in [T]$ . To model the assignment of evacuees from one collection point  $i \in [S]$  to another  $k \in [S]$ , we use variables  $z_{ik} \in \mathbb{B}$ . Finally, to model the bus scheduling part of the problem, we follow [GGH13] and use variables  $x_{ij}^{br} \in \mathbb{B}$  to determine if bus  $b \in [B]$  travels from  $i \in [S]$  to  $j \in [T]$  in round  $r \in [R]$ . The variable  $T$  denotes the total evacuation time, while  $t_{to}^{br}$  and  $t_{back}^{br}$  are auxiliary variables to measure bus travel times.

$$\begin{aligned}
& \min T \\
& \text{s.t. } T \geq \sum_{r \in [R]} (t_{to}^{br} + t_{back}^{br}) + \sum_{i \in [S]} \sum_{j \in [T]} d_i^{start} x_{ij}^{b1} & \forall b \in [B] \quad (1) \\
& t_{to}^{br} = \sum_{i \in [S]} \sum_{j \in [T]} d_{ij} x_{ij}^{br} & \forall b \in [B], r \in [R] \quad (2) \\
& t_{back}^{br} \geq d_{ij} \left( \sum_{k \in [S]} x_{kj}^{br} + \sum_{l \in [T]} x_{il}^{b,r+1} - 1 \right) & \forall b \in [B], r \in [R-1], i \in [S], j \in [T] \quad (3) \\
& \sum_{i \in [S]} \sum_{j \in [T]} x_{ij}^{br} \leq 1 & \forall b \in [B], r \in [R] \quad (4) \\
& \sum_{i \in [S]} \sum_{j \in [T]} x_{ij}^{br} \geq \sum_{i \in [S]} \sum_{j \in [T]} x_{ij}^{b,r+1} & \forall b \in [B], r \in [R-1] \quad (5) \\
& \sum_{j \in [T]} \sum_{b \in [B]} \sum_{r \in [R]} x_{ij}^{br} \geq \sum_{k \in [S]} l_k z_{ki} & \forall i \in [S] \quad (6) \\
& \sum_{i \in [S]} \sum_{b \in [B]} \sum_{r \in [R]} x_{ij}^{br} \leq u_j y_j^T & \forall j \in [T] \quad (7) \\
& \sum_{j \in [T]} y_j^T \leq N^{\text{shelter}} & (8) \\
& \sum_{i \in [S]} y_i^S = N^{\text{coll}} & (9) \\
& \sum_{k \in [S]} l_k z_{ki} \leq L_i y_i^S & \forall i \in [S] \quad (10) \\
& \sum_{k \in [S]} z_{ik} = 1 & \forall i \in [S] \quad (11) \\
& t_{ki}^{\text{walk}} z_{ki} \leq t^{\text{maxwalk}} & \forall i, k \in [S] \quad (12) \\
& x_{ij}^{br} \in \mathbb{B} & \forall i \in [S], j \in [T], b \in [B], r \in [R] \quad (13) \\
& t_{to}^{br}, t_{back}^{br} \in \mathbb{R} & \forall b \in [B], r \in [R] \quad (14) \\
& y_i^S \in \mathbb{B} & \forall i \in [S] \quad (15) \\
& y_j^T \in \mathbb{B} & \forall j \in [T] \quad (16) \\
& z_{ik} \in \mathbb{B} & \forall i, k \in [S] \quad (17) \\
& T \in \mathbb{R}^+ & (18)
\end{aligned}$$

Constraints (1) to (7) are used to model the bus scheduling aspect. Constraint (1) sets the total evacuation time as the maximum of the travel time

over all buses. Constraints (2) and (3) determine the travel times from collection points to shelters and back again, respectively. Constraint (4) ensures that only one journey can be made per round, and connectivity is ensured by Constraint (5). By (6) and (7) we model that all evacuees need to be transported to shelters, and shelter capacities are respected. Note that a shelter  $j$  we do not use ( $y_j^T = 0$ ) is handled as having capacity 0.

The following Constraints (8) to (12) are used to model the location planning aspects. Constraints (8) and (9) ensure that at most  $N^{\text{shelter}}$  shelters can be used, and  $N^{\text{coll}}$  collection points, respectively. Constraint (10) is used to model that we can only assign evacuees to a collection point if that point is open; and if it is open, the capacity needs to be respected. We model that every group of evacuees needs to be assigned to some collection point using Constraint (11). Finally, Constraint (12) ensures that assignments with overlength walking times are forbidden.

### 3 Column Generation

Column generation (see, e.g., [DL05]) has been successfully applied to many problems in vehicle scheduling [NS07, PDH08]. In the following, we describe an approach to generate bus routes for the IBEP as a part of a branch-and-bound scheme.

#### 3.1 Route Reformulation

We represent a route  $p$  in the following reformulation using these aspects:

- A vector  $\mathcal{L}_p \in \mathbb{N}^S$  denoting how many evacuees are picked up at collection points.
- A vector  $\mathcal{U}_p \in \mathbb{N}^T$  denoting how many evacuees are brought to shelter locations.
- The route duration  $t_p$ .

For a given set of routes  $\mathcal{P}$ , we reformulate the IBEP as:

$$\min T \tag{19}$$

$$\text{s.t. } T \geq \sum_{p \in \mathcal{P}} t_p \lambda_p^b \quad \forall b \in [B] \tag{20}$$

$$\sum_{p \in \mathcal{P}} \sum_{b \in [B]} \mathcal{L}_{pi} \lambda_p^b \geq \sum_{k \in [S]} l_k z_{ki} \quad \forall i \in [S] \tag{21}$$

$$\sum_{p \in \mathcal{P}} \sum_{b \in [B]} \mathcal{U}_{pj} \lambda_p^b \leq u_j y_j^T \quad \forall j \in [T] \tag{22}$$

$$\sum_{p \in \mathcal{P}} \lambda_p^b \leq 1 \quad \forall b \in [B] \tag{23}$$

$$\sum_{j \in [T]} y_j^T \leq N^{\text{shelter}} \tag{24}$$

$$\sum_{i \in [S]} y_i^S = N^{\text{coll}} \tag{25}$$

$$\sum_{k \in [S]} l_k z_{ki} \leq L_i y_i^S \quad \forall i \in [S] \quad (26)$$

$$\sum_{k \in [S]} z_{ik} = 1 \quad \forall i \in [S] \quad (27)$$

$$t_{ki}^{\text{walk}} z_{ki} \leq t^{\text{maxwalk}} \quad \forall i, k \in [S] \quad (28)$$

$$y_j^T \leq 1 \quad \forall j \in [T] \quad (29)$$

$$y_i^S \leq 1 \quad \forall i \in [S] \quad (30)$$

$$z_{ki} \leq 1 \quad \forall i, k \in [S] \quad (31)$$

$$\lambda_p^b \in \mathbb{R}^+ \quad \forall b \in [B], p \in \mathcal{P} \quad (32)$$

$$y_j \in \mathbb{R}^+ \quad \forall j \in [T] \quad (33)$$

$$T \in \mathbb{R}^+ \quad (34)$$

Here,  $\lambda$  represents a convex combination of paths for every bus  $b$ . To ensure feasibility, we add an ideal path  $p_0$  at the beginning of the column generation process. For  $p_0$ , we set  $\mathcal{L}_{p_0} = (L_1, \dots, L_S)$ ,  $\mathcal{U}_{p_0} = (0, \dots, 0)$ , i.e., the maximum possible number of evacuees is picked up at every collection point, and no shelter capacity is used. The length of route  $p_0$  is a sufficiently large constant  $M$ .

### 3.2 Pricing

For a bus  $b$ , we construct the following directed graph  $G = (V, A)$  to model trips between collection points and shelters: We define the set of nodes as

$$\begin{aligned} V &= V^S \cup V^T \cup \{s, t\} \\ \text{where } V^S &= \{v_{ir}^S : i \in [S], r \in [R]\}, \\ V^T &= \{v_{jr}^T : j \in [T], r \in [R]\}, \end{aligned}$$

i.e., we expand the collection points and shelters by the maximum number of trips, and add a start node  $s$  representing the depot, and a node  $t$  representing the end of the evacuation for bus  $b$ . These nodes are connected in the following way:

$$\begin{aligned} A &= A^s \cup A^{ST} \cup A^{TS} \cup A^t, \\ \text{where } A^s &= \{(s, v_{i0}^S) : i \in [S]\} \\ A^{ST} &= \{(v_{ir}^S, v_{jr}^T) : i \in [S], j \in [T], r \in [R]\} \\ A^{TS} &= \{(v_{jr}^T, v_{i,r+1}^S) : i \in [S], j \in [T], r \in [R-1]\} \\ A^t &= \{(v_{jr}^T, t) : j \in [T], r \in [R]\} \cup \{(s, t)\}, \end{aligned}$$

where  $A^{ST}$  and  $A^{TS}$  model the possible trips from collection points to shelters and back as complete bipartite graphs, and  $A^s$  and  $A^t$  are starting and ending arcs, respectively.

An example for such a graph is given in Figure 2, where  $R = 2$ , the node  $s$  is on the left side, and node  $t$  as well as arcs  $A^t$  are left out for the sake of clearness.

The pricing problem consists of finding a path in that graph (that is, a new route for bus  $b$ ) using the reduced costs of the current optimal solution.

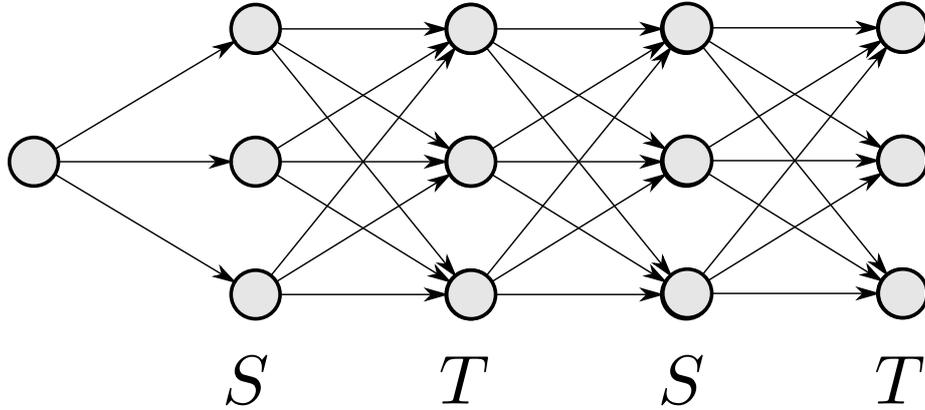


Figure 2: Example for  $G$ .

Let  $\pi_b^{(20)}$ ,  $\pi_i^{(21)}$  and  $\pi_j^{(22)}$  be the corresponding dual variables of the respective constraints. We solve the following problem:

$$\min \pi_b^{(20)} t_p - \sum_{i \in [S]} \pi_i^{(21)} \mathcal{L}_{pi} - \sum_{j \in [T]} \pi_j^{(22)} \mathcal{U}_{pj} - \pi_b^{(23)} \quad (35)$$

$$\text{s.t. } p \text{ is a path from } s \text{ to } t \text{ in } G, \quad (36)$$

Recall that  $\mathcal{L}$  and  $\mathcal{U}$  denote how often a collection point or shelter is travelled to, respectively. This can be easily counted by moving these costs to the preceding or subsequent arc; i.e., we only need to solve a shortest path problem in the pricing step, with arc lengths

$$\begin{aligned} \pi_b^{(20)} d_i^{\text{start}} & \text{ for } a = (s, v_{i0}^S) \in A^s \\ \pi_b^{(20)} d_{ij} - \pi_i^{(21)} - \pi_j^{(22)} & \text{ for } a = (v_{ir}^S, v_{jr}^T) \in A^{ST} \\ \pi_b^{(20)} d_{ij} & \text{ for } a = (v_{jr}^T, v_{ir+1}^S) \in A^{TS} \\ -\pi_b^{(23)} & \text{ for } a \in A^t \end{aligned}$$

which can be done efficiently.

## 4 Branch and Bound

### 4.1 Starting Solution

As it would be possible to start the column generation process using only the ideal route  $p_0$ , a starting solution does not necessarily need to be provided. However, having an upper bound available from the beginning improves the subsequent branching steps.

To calculate a starting solution, we find a feasible solution to the integrated IP model sequentially: We first determine a heuristic set of shelter locations, then a heuristic set of collection points with evacuee assignments, and finally a heuristic set of bus routes.

For the first step, the set of shelter locations, we solve the following small IP:

$$\min \sum_{j \in [T]} d_j^{av} y_j \quad (37)$$

$$\text{s.t.} \quad \sum_{j \in [T]} u_j y_j \geq \sum_{i \in [S]} l_i \quad (38)$$

$$\sum_{j \in [T]} y_j \leq N^{\text{shelter}} \quad (39)$$

$$y_j \in \mathbb{B} \quad \forall j \in [T], \quad (40)$$

where  $d_j^{av} = \sum_{i \in [S]} d_{ij} l_i$  denotes the weighted distance to all collection points. In other words, we choose a set of shelters with cardinality of at most  $N^{\text{shelter}}$  that provide sufficient capacity to accommodate all evacuees, such that these shelters are as close to the collection points as possible.

In a subsequent second step, we find a set of collection points and assignments. Again, this is done by solving a small IP:

$$\min \sum_{i \in [S]} d_i^{av} x_i \quad (41)$$

$$\text{s.t.} \quad \sum_{k \in [S]} l_k z_{ki} \leq L_i x_i \quad \forall i \in [S] \quad (42)$$

$$\sum_{i \in [S]} x_i \leq N^{\text{coll}} \quad (43)$$

$$\sum_{k \in [S]} z_{ik} = 1 \quad \forall i \in [S] \quad (44)$$

$$t_{ik}^{\text{walk}} z_{ik} \leq t^{\text{maxwalk}} \quad \forall i, k \in [S] \quad (45)$$

$$x_i \in \mathbb{B} \quad \forall i \in [S] \quad (46)$$

$$z_{ik} \in \mathbb{B} \quad \forall i, k \in [S], \quad (47)$$

where  $d_i^{av} = \sum_{j \in T'} d_{ij}$  and  $T' \subseteq [T]$  denotes the chosen shelters from the preceding step.

Using this choice of collection point and shelter decision, we finally generate a feasible bus schedule. To do so, we reuse the branch and bound algorithm from [GGH13] with a bound on the maximum runtime.

## 4.2 Branching Order

Having generated a feasible starting solution that provides us with an upper bound, and having solved the root problem using the column generation approach described in Section 3 that provides us with a lower bound, we begin the branching process. For the currently considered node  $n$  within the branching tree, we proceed as follows:

1. If one of the variables  $y_j^T$  is not fixed to its lower or upper bound, create two new problems with either  $y_j^T = 0$  and  $y_j^T = 1$ , respectively. Remove  $n$  from the set of active nodes. Always begin with the variable  $y_j^T$  for which the value  $u_j$  is the largest.

2. If all  $y^T$  variables are fixed, and one of the variables  $y_i^S$  is not fixed to its lower or upper bound, branch on this variable as in 1. Always begin with the variable  $y_i^S$ , for which the value  $L_i$  is the largest.
3. If both  $y^S$  and  $y^T$  are fixed, branch on the assignment decisions  $z_{ki}$ , using that only one of the values  $z_{ki}$ ,  $k \in [S]$ , can be one.
4. If all decisions  $y^S$ ,  $y^T$  and  $z$  are fixed, reuse the branch and bound algorithm from [GGH13] to solve the resulting BEP.

Whenever we branch a node, we solve the linear relaxation of the resulting subproblems using column generation. In the next step, we choose one of the active nodes with smallest lower bound and iterate the branching process. Whenever a BEP subproblem is solved, we may improve our current best solution and thus the upper bound. In a pruning step we remove all nodes whose lower bound is at least as large as the current upper bound.

### 4.3 Additional cuts

During the branching process, we can add cuts to the route formulation of the IBEP based on the shelter capacities. As an example for the basic idea of these cuts, consider the following problem: There remain three busloads to be evacuated, and two possible shelter locations, of which only one can be chosen: Shelter 1 with a capacity of 2, which is close to the endangered area; and shelter 2 with a capacity of 4, which is farther away.

An optimal solution to the relaxed problem might choose to open both shelters 1 and 2 with half capacity (i.e., to set  $y_1^T = y_2^T = 0.5$ ). This results in a sufficient total capacity of  $0.5 \cdot 2 + 0.5 \cdot 4 = 3$ , and the fast evacuation to shelter 1 can be used. However, any integral solution cannot open shelter 1, as this alone does not provide sufficient shelter capacity for the three remaining busloads. Thus, we can deduce that  $y_2^T = 1$  needs to be fulfilled.

Formally, we apply the following cutting procedure to every node within the branch-and-bound tree (for the sake of simplicity, we use a notation as if we were considering the root node):

- (1.) Solve the relaxed problem formulation. Let  $T'$  be the set of shelters that are used, i.e.,  $T' = \{j \in [T] : y_j^T > 0\}$ . Note that  $|T'| \geq N^{\text{shelter}}$ .
- (2.) For every subset  $T'' \subseteq T'$  with  $|T''| = N^{\text{shelter}}$ , do: If  $\sum_{j \in T''} u_j \leq \sum_{k \in [S]} l_k$ , then add the cut  $\sum_{j \in T''} y_j^T \leq N^{\text{shelter}} - 1$ .
- (3.) Resolve, and return to (2.) until no more cuts can be added.

Note that in practice,  $|T'| - N^{\text{shelter}}$  will be small, and thus Step (2.) can be checked quickly.

## 5 Experimental Results

**Environment** All experiments were conducted on an Intel Xeon E5-2670 machine with 16 cores at 2.6 GHz and 32GB of main memory, running under Ubuntu. All programs were pinned to one core. Code was compiled using gcc

4.6.3 with optimization flag -O3. Linear programs were solved using CPLEX 12.4, and shortest path problems using LEMON 1.2.3<sup>1</sup>.

## 5.1 Randomized instances

**Datasets** We generated two sets of instances for this experiment:

1.  $\mathcal{I}_1(S, T, B, N^{\text{coll}}, N^{\text{shelter}})$ , sets with completely random values: Distances  $d_{ij}$  are chosen randomly from  $\{1, \dots, 5\}$ , numbers of evacuees  $l_i$  at collection points from  $\{1, \dots, 15\}$ , collection point capacities  $L_i$  from  $\{l_i + 1, \dots, l_i + 10\}$ , and capacities at shelters from  $\{1, \dots, 20\}$ .
2.  $\mathcal{I}_2(S, T, B, N^{\text{coll}}, N^{\text{shelter}})$ , instances with Euclidean distances: We generate locations randomly in a plane, and generate travel times using the Euclidean distance. Specifically, we assumed 5 concentric zones, see Figure 3. Within zone 1 (radius 2-5),  $S$  collection points are generated with  $l_i \in \{1, \dots, 15\}$  and  $L_i \in \{l_i + 1, \dots, l_i + 10\}$ . Note that we do not allow points that are too close to the center, to ensure some minimum diversity. In zones 2 (radius 10-15), 3 (radius 15-20), and 4 (radius 20-25) we generate shelters with capacities  $u_j$  from  $\{1, \dots, 5\}$ ,  $\{1, \dots, 10\}$ , and  $\{1, \dots, 20\}$ , respectively. This means that shelters that are farther away from the center may have larger capacities. The maximum walking distance is 5 (the radius of zone 1).

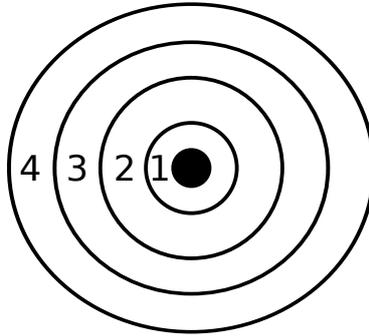


Figure 3: Zones for generating instance set  $\mathcal{I}_2$ .

For  $\mathcal{I}_1$  and  $\mathcal{I}_2$ , we used the sets of parameters as presented in Table 1. For each parameter set, we generated 10 instances, dropping infeasible ones. Thus, we created 240 instances in total.

**Setup** For each instance, we solve the integrated model using Cplex and our branch-price-and-cut approach. Computation times were restricted to 20 minutes per instance, and memory usage to 10 GB. Furthermore, we note the objective value of the sequential starting solution used for the branch-price-and-cut approach.

<sup>1</sup><http://lemon.cs.elte.hu/trac/lemon>

$S$	$T$	$B$	$N^{\text{coll}}$	$N^{\text{shelter}}$
4	4	3	2	2
4	4	3	3	3
5	5	3	2	2
5	5	3	3	3
5	5	3	4	4
6	6	4	2	2
6	6	4	3	3
6	6	4	4	4
7	7	4	3	3
7	7	4	4	4
8	8	4	3	3
8	8	4	4	4

Table 1: Instance parameters.

**Results** We show the average gap  $(UB - LB)/LB$  over the ten instances of each type in percent and the number of instances where a proven optimum was found in Tables 2 and 3. The tables show that our BCP approach is able to find a considerably smaller gap than Cplex, and solves more instances to optimality. The performance on both instance types  $\mathcal{I}_1$  and  $\mathcal{I}_2$  is approximately the same.

$S$	$T$	$B$	$N^{\text{coll}}$	$N^{\text{shelter}}$	Cplex		BCP	
					OPT	GAP	OPT	GAP
4	4	3	2	2	4	10.57	10	0.00
4	4	3	3	3	0	39.77	9	0.14
5	5	3	2	2	1	30.92	8	0.36
5	5	3	3	3	0	42.98	6	1.57
5	5	3	4	4	0	45.13	4	3.77
6	6	4	2	2	0	33.98	9	0.16
6	6	4	3	3	0	45.82	5	3.69
6	6	4	4	4	0	49.28	7	3.50
7	7	4	3	3	0	48.26	3	2.46
7	7	4	4	4	0	51.63	3	4.23
8	8	4	3	3	0	49.87	3	3.48
8	8	4	4	4	0	51.48	2	5.39

Table 2: Results for  $\mathcal{I}_1$ . OPT denotes the number of instances for which a proven optimum was found. GAP denotes the average IP gap in percent.

We compare the best solutions found within the given time and memory limit in Tables 4 and 5. The numbers show the average loss in quality compared to the respective best solution, i.e.,  $UB_i/UB_{best} - 1$ , in percent. Note that the heuristic solution (the starting solution for the BCP, calculated by the sequential approach) only takes about up to 5 seconds to compute, while the other approaches have up to 1200 seconds.

The results show that a) the BCP finds the superior gap not only by better lower bound computations, but also by a better feasible solution; and b) the sequential approach is better on instances of type  $\mathcal{I}_2$ . That is to be expected, as the existence of metric distances is implicitly assumed when average distances

$S$	$T$	$B$	$N^{\text{coll}}$	$N^{\text{shelter}}$	Cplex		BCP	
					OPT	GAP	OPT	GAP
4	4	3	2	2	1	21.06	8	1.20
4	4	3	3	3	0	42.30	5	1.75
5	5	3	2	2	0	26.31	9	0.30
5	5	3	3	3	0	46.25	4	1.48
5	5	3	4	4	0	45.29	3	2.26
6	6	4	2	2	0	35.05	7	0.96
6	6	4	3	3	0	45.62	3	1.96
6	6	4	4	4	0	48.41	2	2.39
7	7	4	3	3	0	46.45	3	3.48
7	7	4	4	4	0	49.13	2	2.80
8	8	4	3	3	0	46.85	2	2.53
8	8	4	4	4	0	(48.67)	0	2.68

Table 3: Results for  $\mathcal{I}_2$ . OPT denotes the number of instances for which a proven optimum was found. GAP denotes the average IP gap in percent. Brackets () indicate that no feasible solution was found for at least one instance.

$S$	$T$	$B$	$N^{\text{coll}}$	$N^{\text{shelter}}$	Cplex	Heu	BCP
4	4	3	2	2	0.00	8.57	0.00
4	4	3	3	3	3.97	23.43	0.00
5	5	3	2	2	0.27	9.32	0.00
5	5	3	3	3	2.03	26.03	0.00
5	5	3	4	4	3.14	19.28	0.71
6	6	4	2	2	1.91	22.22	0.00
6	6	4	3	3	4.10	11.19	0.00
6	6	4	4	4	7.27	13.98	0.00
7	7	4	3	3	5.70	9.39	0.00
7	7	4	4	4	10.41	23.52	0.00
8	8	4	3	3	4.72	14.42	0.00
8	8	4	4	4	8.25	9.33	0.22

Table 4: Normalized best solutions for  $\mathcal{I}_1$ .

$S$	$T$	$B$	$N^{\text{coll}}$	$N^{\text{shelter}}$	Cplex	Heu	BCP
4	4	3	2	2	0.37	0.43	0.00
4	4	3	3	3	0.52	3.28	0.23
5	5	3	2	2	0.45	9.07	0.00
5	5	3	3	3	0.53	3.37	0.00
5	5	3	4	4	1.13	2.90	0.00
6	6	4	2	2	0.49	2.39	0.00
6	6	4	3	3	2.07	6.28	0.00
6	6	4	4	4	3.00	4.34	0.00
7	7	4	3	3	0.88	3.47	0.45
7	7	4	4	4	4.62	5.83	0.00
8	8	4	3	3	2.01	4.34	0.30
8	8	4	4	4	(1.81)	4.49	0.00

Table 5: Normalized best solutions for  $\mathcal{I}_2$ . Brackets () indicate that no feasible solution was found for at least one instance.

are calculated (see Section 4.1).

## 5.2 Kaiserslautern

In our second experiment, we consider the real-world instance of Kaiserslautern, Germany, as a case study. We assume a similar scenario as in [GG12]: Within a  $500m$ -radius, the city center needs to be evacuated, caused by a bomb disposal as an example. There are 14 bus stops within the affected region, which are used as the set of possible gathering points for evacuees. Bus stop capacities are estimated based on the size of open area surrounding a stop. Furthermore, we identified 23 gymnasiums within the vicinity that may serve as evacuee shelters, and estimate their capacity by using their area. Figure 4 visualizes this scenario, using squares for bus stops, and triangles for gymnasiums.

Travel distances are calculated using the Google Distance Matrix API, using either walking speed for bus stop – bus stop distances, or driving speed for gymnasium – bus stop distances. Using the population density of the city center, we estimate a total of 21 bus loads of passengers that need to be evacuated.

Finally, for these considerations we assume a maximum walking distance of 7 minutes for each evacuee, and having three buses available.

We are now able to calculate the estimated evacuation time for different numbers of collection points and shelters. Using the BCP algorithm, calculated evacuation times in minutes are presented in Table 6. The left number is the respective lower bound, and the right number the upper bound in every scenario. This allows a planner to try different evacuation approaches, and see directly their impact on evacuation times. As an example, 10 minutes may be saved when the number of used shelters is increased from 3 to 6 for  $N^{\text{coll}} = 3$ .

		$N^{\text{shelter}}$			
		3	4	5	6
$N^{\text{coll}}$	3	51/51	47/47	42/42	41/41
	4	45/45	41/41	38/41	38/38

Table 6: Objective values for Kaiserslautern instance with different choices for  $N^{\text{shelter}}$  and  $N^{\text{coll}}$ . Left number denotes the lower bound, right number the upper bound.



Figure 4: Kaiserslautern instance. (image copyright 2012 Google and 2012 GeoBasis)

## 6 Conclusion

Location-routing problems integrate a location planning aspect with a vehicle routing problem, and thus have the potential of finding solutions with superior objective value than a sequential approach. However, computational effort increases.

In this paper we presented a location-routing model for evacuating a region with the help of buses. The locations of both gathering and shelter points need to be determined, along with a bus schedule. We developed a branch-cut-and-price strategy, in which the pricing problem is a shortest path problem in a round-expanded graph. Our computational experiments show that it is possible to solve realistically-sized instances to optimality this way, and that a commercial IP solver is considerably outperformed.

## References

- [AGI06] N. Altay and W. G. Green III. OR/MS research in disaster operations management. *European Journal of Operational Research*, 175:475–493, 2006.
- [BBP<sup>+</sup>11] J.-M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. W. Calvo. A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Research*, 38(6):931 – 941, 2011.
- [Bis11] D. R. Bish. Planning for a bus-based evacuation. *OR Spectrum*, 33:629–654, 2011.
- [CCB01] Y. Chan, . B. Carter, and M. D. Burnes. A multiple-depot, multiple-vehicle, location-routing problem with stochastically processed demands. *Computers & Operations Research*, 28(8):803 – 826, 2001.
- [DL05] Jacques Desrosiers and MarcoE. Lübbecke. A primer in column generation. In Guy Desaulniers, Jacques Desrosiers, and MariusM. Solomon, editors, *Column Generation*, pages 1–32. Springer US, 2005.
- [GG12] M. Goerigk and B. Grün. The robust bus evacuation problem. Technical report, Fachbereich Mathematik, Technical University of Kaiserslautern, 2012.
- [GGH13] M. Goerigk, B. Grün, and Ph. Heßler. Branch and bound algorithms for the bus evacuation problem. Technical report, Fachbereich Mathematik, Technical University of Kaiserslautern, 2013.
- [NS07] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177:649–672, 2007.
- [PDH08] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.