Road network segmentation for bus-based evacuation

Md. Imran Hossain
University of the Bundeswehr Munich
Institute for Applied Computer Science
Werner-Heisenberg-Weg 39
85577 Neubiberg, Germany
Imran.Hossain@unibw.de

Wolfgang Reinhardt
University of the Bundeswehr Munich
Institute for Applied Computer Science
Werner-Heisenberg-Weg 39
85577 Neubiberg, Germany
Wolfgang.Reinhardt@unibw.de

Abstract

Mass evacuation of large groups of people is a very important task in some specific cases of disasters. In these cases of evacuation a number of vehicles (buses) are used to collect the people living in a specific area and to transfer them to a safe place or a hospital, often under a time constraint. As in general there is a large number of people and a limited number of evacuation units (vehicles/buses) with limited capacities the optimization of the evacuation related to routes and time is a complex problem. In literature this optimization problem is often called the bus based evacuation problem (BBEP). BBEP algorithms assume that the number of evacuees in a specific area is given and that they can be collected from predefined fixed locations (source points). Such predefined fixed source points certainly limit the evacuation process. Therefore in this paper we present an extension of the published evacuation approaches by introducing a way to overcome the limitations of predefined fixed locations and to collect the people from the premises where they live instead.

Keywords: Assisted Evacuation, Bus-based Evacuation Problem (BBEP), Spatial Algorithm, Source Points generation

1 Introduction

When a massive destruction is projected for a forthcoming disaster, evacuation often become so obvious to minimize the casualties. In some cases, the responsible authorities (public safety agencies, police department etc.) provide special evacuation units (vehicles) to collect and shift the people to a safe place which is called herein as the assisted evacuation. The route plan for each evacuation unit has significant effects on the efficiency of assisted evacuation.

Quite a large number of operation researches have been carried out in the domain of evacuation management mainly focusing on the self-evacuation or car based evacuation. The authors refer to [3, 4, 9, 10, 11] for some good models and algorithms on car-based evacuation problems. However, very few researches address the issue of assisted evacuation. Recently, a few number of researches came out with transit-based models that are pretty close the assisted evacuation problem. We have found the bus-based evacuation planning problem (BBEP) developed by Bish [2] as a closest solution for the assisted evacuation problem.

Bish considered the BBEP identical to the Vehicle Routing Problem (VRP) [1]. For the BBEP, a number of depots, where initially a number of buses with defined capacity are situated, a number of pickup locations (sources) where evacuees should be gathered for their transit, a number of safe distinctions (shelters or sinks) with available minimum requirements for evacuees like beds, blankets, food, etc. are given [12]. The number of evacuees at the sources and the capacities at the sinks are known in advance. The problem that BBEP solves is to transport evacuees from sources to sinks in the minimal amount of time, i.e., the minimal duration of evacuation by routing and scheduling a set of homogeneous and capacitated buses. The duration of evacuation is defined as the time span between when the first bus leaves its depot until the last evacuees is reached to the sinks [12].

The BBEP is further enriched and modified by Goerigk, Grün, and Heßler [6] by developing branch and bound algorithms. Goerigk and Grün [5] further extended the BBEP to a robust bus-based evacuation problem (RBBEP) assuming that the number of evacuees are unknown while setting estimation of evacuees at each source point. For other extensions we refer to [7, 8].

In general the BBEP/RBBEP is a very good approach to optimize the assisted evacuation process if we consider that the source points are fixed and well known in advance both to the evacuees and the bus operator and the evacuees are able to gather themselves in those source points. Now the problem is, first of all, the BBEP/RBBEP does not tell anything about how to create/define those source points. Secondly, a non-optimized set of source points could hinder the total optimization process. And finally, in reality, there exists evacuees who are subject to severe mobility restriction and thus cannot arrange themselves move to the source points. Therefore, a door to door collection of the evacuees with automatic optimized set of source points certainly preside over the given fixed source base collection of BBEP. For example while collecting evacuees from door to door a source point has to be created at the point where the evacuation unit/bus is full. If the source points could be created in such a way the general BBEP could be applied afterwards considering those source points and given number of sinks.

Our contribution: In this paper, we focus on the approach of door to door collection of evacuees. We introduce an approach followed by an algorithm to create the source points dynamically depending on the location of the evacuees and the capacity of the evacuation units. The algorithm is implemented and applied to a test scenario and the results are discussed.

2 An approach for automatic source points creation

2.1 Problem definition

Input: Let $B = \{b_1,\dots, b_n\} \mid n \in \mathbb{N}$, be the set of buildings and $R = \{r_1,\dots, r_m\} \mid m \in \mathbb{N}$, be the set of the roads in a road network. It is normal that the number of buildings in a given area is higher than the number of roads and we assume that $n > m$. Each building $b_i \in B \mid 1 \le i \le n$, has given a number of evacuees $E(b_i)$ and name of the access road $M(b_i)$. Each road $r_j \in R \mid 1 \le j \le m$, has a given an address including the name $M(r_j)$ and a given direction vector $V(r_j)$. Let C be the capacity of an evacuation unit/bus. We assume that C is the same for all evacuation units/buses.

Output: Segmentation of each $r_j \in R$ into a set like $\{ (r_j)_1,\dots, (r_j)_k \} \mid 0 \le k$, based on the criteria of the equation 1. The $E(r_j)_p \mid p \le k$, in equation 1 is the value of an element $(r_j)_p \in r_j$ that refer to the number of evacuees associated with $(r_j)_p$. In other word, each small segment of a road should consist of a number of evacuees equal to the capacity of an evacuation unit. Besides, a set of source points $S = \{s_1,\dots, s_l\} \mid l \in \mathbb{N}$ has to be created satisfying the equation 2. Also in other words each small segment of a road has to be given a source point.

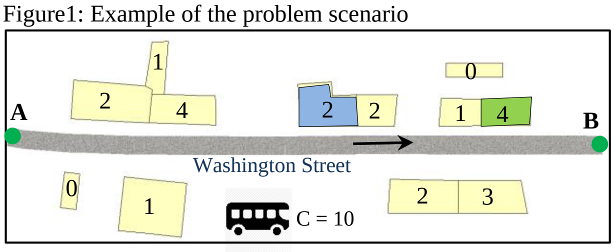
$$E(r_j)_p = C\dots \qquad (1)$$

$$l = \sum_{1}^{m} \sum_{1}^{k} r_{mk} \forall r = 1 \dots \qquad (2)$$

Exception: only one sub-segment of a road can violate criteria mentioned in equation 1 where $E(r_j)_p$ could be less than C as the number of evacuees living in a road is not a multiple of the capacity of an evacuation unit.

Example: The problem and the expected solution could be illustrated best, as it is often the case, with pictures. In figure 1 the polygons represent the buildings and the number inside each polygon represents the number of evacuees. A single road "Washington Street" which starts at point *A* and finished at point *B* is represented by the gray thick line. The arrow inside the road shows the direction by which a vehicle can traverse. Let us assume that the capacity of the evacuation unit/bus is 10.

Figure1: Example of the problem scenario



Now imagine that the evacuation units starts its operation from point *A*. The vehicle will get full at the blue building and at this point a source point *C* has to be created from where the vehicle directly goes to the sink. In the next round just prior to the green building there will be 8 evacuees inside the vehicle. Therefore, it can take only 2 evacuees from the green building and leave the other 2 for the next round creating a source point *D*. Thus, the last segment DB would contain only 3 evacuees and the source point *e* lies somewhere in the segment.

Figure 2: Expected solution of the problem

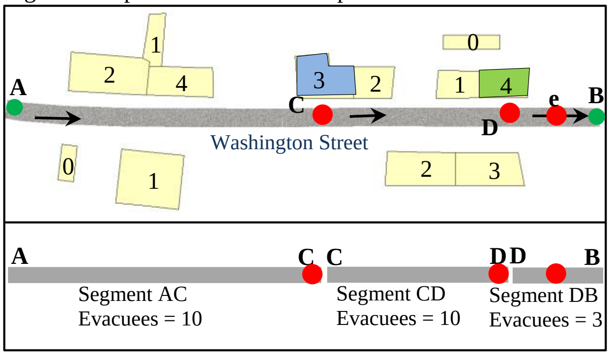


Figure 2, which represents the outputs, shows that the Washington Street AB is divided into 3 parts: AC, CD and DB. Except the segment DB the other segments contains exactly 10 evacuees. In addition, 3 source points (red circles) have been created for all segments. The segment DB which contains 3 evacuees could be combined with the other road (if exist) linked with Washington Street at the point B to further optimize the process but this issue is treated as a future work.

2.2 The algorithm

The input and output for the algorithm have already been described in section 2.1. In general, the algorithm first transfers the number of evacuees from a given building layer to the given associated road network layer, segments the road network layer in a way that each segment contains the number of evacuees equal to the evacuation unit capacity and finally creates a source point on each segment in a defined specific location. To avoid complexity in this initial attempt we assume that the capacity of all evacuation units are identical. The algorithm is described in more detail through the following three steps with pseudo codes and graphics.

Step 1: Transfer of no. of evacuees from building to road: The algorithm (pseudo codes) and a graphical illustration of this step are given in the figure 3.a and figure 3.b respectively. In this very first step the number of evacuees $E(b)$ reside in each buildings b is projected to its associated road r in a certain point ("Road Point" p) on the road r. A match between the road name from the building's address and the road names of roads figures out the road to which a building is connected. This approach eliminates the dilemma of associated/connected road for a building located at a junction of roads.

To do this projection, at first, the algorithm creates an empty point set $P = \{\}$ for road points. Then for each building $b \in B$ the algorithm searches the road $r \in R$ to which it is connected. This is done by matching the road name $M(r)$ of r with the road name $M(b)$ from the building address. Then the centroids ($centroid(x,y)$) of the building is calculated and a perpendicular line pL is drawn from the $centroid(x,y)$ to the r. A point $p \in P$ is then created at the intersection point between r and pL. The no. of evacuees of the building b and some attributes of the r and b like "building ID", "Road ID" are stored with the road point p in addition. The p is then added to the set of P.

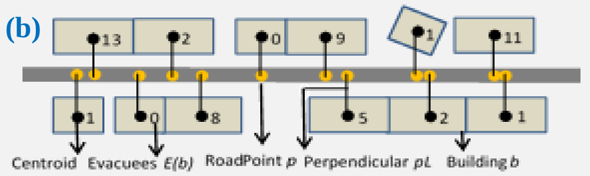
Figure 3: Algorithm subset and graphical illustration of step 1

Data: FeatureSet *B*, FeatureSet *R* and No. of Evacuees attribute *E(b)*
Result: FeatureSet *RoadPoints P*

(a)

```
/*creating an empty point FeatureSet for RoadPoints with
  necessary attributes like "No of evacuees", "building ID"
  "raod ID" etc.
Create RoadPointsSet P ← ∅
/* populating the RoadPoints FeatureSet
for each Building b ∈ B do
    /*function for finding the road which name is same to the
      road name in building address. This returns the road r
    function [FindAssociatedRoad (M(b))] | returns road r ∈ R
    /*this function calculates and returns the centroid point
      of the building.
    function [CalculateCentroidPoint (b)] | returns centroid (x,y)
    /*this function calculates and returns a perpendicular line from
      centroid (x,y) to the found road r
    function [DrawPerpendicularLine (from centroind (x,y) to r )]
                 | returns perpendicularLine pL
    /*this function calculates and returns a point p ∈ P at the
      intersection of r and pL
    function [CreatePointAtIntersection (r, pL)] | returns p ∈ P
    /*this function populates the attributes of p ∈ P
    function [PopulateAttributesOf(p)]
                 | returns p ∈ P with attributes
    /*this procedure adds the p to P set
    procedure [Add p to RoadPointsSet P]
return RoadPointsSet P
```

(b)



Step 2: Linear referencing of the elements of the set *P*: The algorithm (pseudo codes) and a graphical illustration of this step are given in the figure 4.a and figure 4.b respectively. The goal of this step is to sort the road points located in each road r according to the distance between the road point p and the start point of the road so that the road points in each road r could be traverse sequentially for creating a source point afterwards.

For this the algorithm first creates one empty set of sets Q for road points and one empty set T for routes. Thereafter, for each road $r \in R$ the r is converted to a route t and added to the T. After that the associated set of road points $SubP \subseteq P$ for r are retrieved through ID. The distance for each element (road point) of *SubP* from the starting point of r is measured with linear referencing technique with the support of the associated route t and is stored with the element as an attribute. The *SubP* is then sorted according to the measured distance in ascending order. At the end the *SubP* is added to the Q set.

Step 3: Road segments and source points creation: The algorithm (pseudo codes) of this step is given in figure 5.a. and is explained with the graphics of the figure 5.b. In this step at first a sorted $SubP \subseteq P$ is retrieved from the set Q. Let us think that the yellow points containing the no. of evacuees in middle in figure 5.b are the elements of a *SubP*. The elements are sorted according to the step 2 and their sequence numbers are given below each element (yellow point). The grey thick line is a single road for which the start point is A and the end Point is B. Let us assume that the capacity of the evacuation unit $C = 10$. The red points with ID inside represent the output source points and the blue lines with ID represent the output road segments. In general, the algorithm sums up the no. of evacuees by traversing through the elements according to the sequence and segments and points are created according to five different conditions.

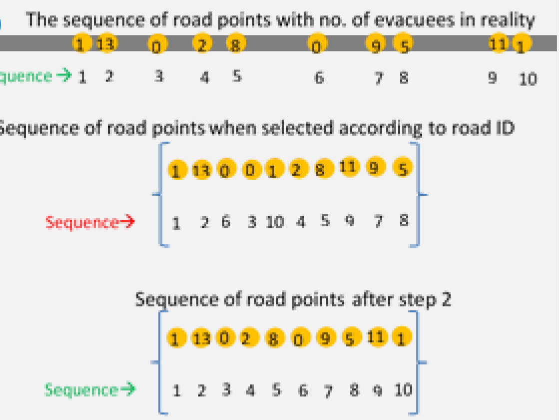
Figure 4: Algorithm subset and graphical illustration of step 2

Data: Road Points FeatureSet *P* and Road FeatureSet *R*
Result: $q \mid q = m$ sorted sets/arrays of Road Points in a set Q and Route FeatureSet *T*

(a)

```
/*creating an empty Route FeatureSet T with attributes similar to
  Road FeatureSet
Create Route T ← ∅
/*creating an empty set of sets Q for q sets of road points
Create Q ← ∅
for each Road r ∈ R do
    /*function for converting the road r to a route t
      This returns the route t of road r
    function [CreateRoute(r)] | returns route t of r
    /*this procedure adds the t to T set
    procedure [Add t to Route T]
    /*select the road points from P that are located on the r
    function [SelectRoadPoints(Road Id = RoadPoint ID)]
    | returns the subset of P as SubP
    /*next loop measures distance from the start point of the
      Road r to each road points in SubP
    for each p ∈ SubP do
        function [MeasureDistance(p to StartPoint of r)
    /*this function returns the SubP with the elements of it sorted
      according to the its measured distance in ascending order
    function [SortSubPAccordingToMeasuredDistance
             (ascending order )] | returns sorted SubP
    /*this procedure adds the SubP to Q set
    procedure [Add SubP to Q]
return Route T
return Q
```

(b)



Condition 1(refers to source point 1 and Segment 1 in figure 5.b): if at any element (ID=2) the sum of evacuees ($SumE$) = C then a source point (ID=1) is created on the location of that element and a road segment (ID=1) is created by splitting the road at the source point. Once a source point is created the value of *SumE* is reset to 0.

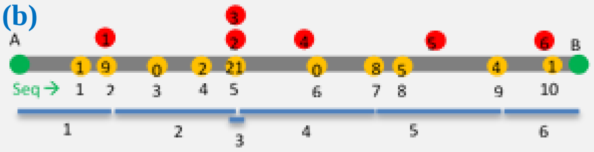
Figure 5: Algorithm subset and graphical illustration of step 3

Data: Q which is the Set of sorted $SubP \subseteq$ P and Evacuation Unit Capacity C
Result: FeatureSet of Source Points S

(a)

```
/*creating an empty point FeatureSet for Source Points P with
  attributes like "No of evacuees" and "road ID"
Create SourcePointSet S ← ∅
/*creating an empty line FeatureSet for Road Segment L with
  attributes like "No of evacuees" and "Segment ID"
Create RoadSegmentSet L ← ∅
/* populating the SourcePointSet and RoadSegmentSet
for each Sorted SubP ∈ Q do
    /*static variable for keeping track evacuees in vehicle
    EvacueesInVehicle EV = 0
    /*static variable for keeping track of remaining evacuees at
      a certain raod point p
    RemainingEvacuees RV = 0
    /*static point variable for keeping track of the start point
      of a raod segment
    StartPoint SP = Start point of road r to which SubP is …
                    associated with
    /*static point variable for keeping track of the end point
      of a raod segment
    EndPoint EP = null
    for each Road Pont p ∈ SubP do
        EV = EV + EvacueesAtRoadPoint p + RV
        if (EV = C) and (RV = 0) do
            function [Create Source Point s (location of
            RoadPoint p) | returns Source Point s
            procedure [Add s to S]
            EP = p
            function [Create Road Segment l (From SP to
            to EP)] | returns Road Segment l
            procedure [Add l to L]
            SP = p, EV = 0, RV = 0
        else if (EV = C) and (RV > 0) do
            function [Create Source Point s (location between
            SP and p) | returns Source Point s
            procedure [Add s to S]
            EP = p
            function [Create Road Segment l (From SP to
            to EP)] | returns Road Segment l
            procedure [Add l to L]
            SP = p, EV = 0, RV = 0
        else if (EV > C) and (RV = 0) do
            function [Create s at location of RoadPoint p…
            u time where u = floor of EV/C
            procedure [Add u no. of s to S]
            EP = p
            function [Create u times Road Segment l
            (From SP to EP)] | returns u Road Segment l
            procedure [Add u no. of l to L]
            SP = p, EV = 0, RV = EV- C*u
        else if (EV > C) and (RV > 0) do
            u = floor of EV/C
            function [Create s for the first instance at location
            Between SP and P and for other instance at p]
            procedure [Add u no. of s to S]
            EP = p
            function [Create u times Road Segment l
            (From SP to EP)] | returns u no. of l
            procedure [Add u no. of l to L]
            SP = p, EV = 0, RV = EV- C*u
return SourcePointSet S
return RoadSegmentSet L
```

(b)



Condition 2 (refers to source point and segment 2 and 3): if the SumE > C at an element (ID=5) and there are no remaining evacuees in the previously created source point (ID=1) then n source points (ID=2,3) are created at that element location where n = rounded lower integer of (SumE)/C. The segment (2) for first source point (ID=2) is created by segmenting the road from previous source point (ID=1) to the location of the current element (ID=5). For the rest of the source points (ID=3), zero lengths segments (ID=3) are created for which the start and end point are the same. Any remaining evacuees (in this case 3) is stored in a variable and the *SumE* reset to 0.

Condition 3 (refers to source point 4 and segment 4): if the *SumE* > *C* at an element (ID=7) and there are remaining evacuees in the previously created source point (ID=3) then a source point (ID=4) at a location in middle between the element (ID=7) and the previous source point (ID=3). A new segment (ID = 4) is created by segmenting the road from previous source point (ID=3) to present element (ID=7). The remaining evacuees (1 evacuees) at the present element is calculated and the *SumE* is reset to 0.

Condition 4(refers to source point 5 and segment 5): if the *SumE* = *C* at an element (ID=9) and there are remaining evacuees in the previously created source point (ID=4) then a source point (ID=5) is created in a middle point between the present element (ID=9) and element (ID=7). Again, a new segment (ID = 5) is created by segmenting the road from element (ID=7) to present element (ID=9).

Condition 5(refers to source point 6 and segment 6): if the *SumE*>0 at the last element (ID=10) then a source point (ID=6) is created in a middle point between the end point of last segment (ID=5) and the end point (B) of the road. A segment (ID=6)is also created between those two points.

3 Result and discussion

The algorithm we presented in section 2 has been implemented using c# and ArcObjects library of ESRI. We have used OpenStreetMap(OSM) data for the input. The addresses of the buildings are generated with geocoding Web Service as the address is not included in the OSM data. The no. of evacuees for buildings are generated randomly. Figure 6 and 7 show the result of 2 examples of the implemented algorithm. The green dots are the source points and the red dots are the road points in both figures. The total no. of evacuees ware 1144 for both cases. In figure 6, with given *C*=25 the algorithm created 57 source points and segments. The number of source points and segments are 26 in the second case with given *C* = 50.

Now if we divide the total evacuees (1144) by two different values of C: 25 and 50 we will get in first case 45 and in second case 22 full loads and some remaining's. This no. of full loads are a bit lower than the actual no. of source points/segments created in figure 6 and 7. In fact this is expected and happens because of the condition 5 of step 3 of the algorithm where a source point in each road is created at the end with the remaining evacuees even though the total no. of evacuees are far less than C.

Point to be noted here that the algorithm so far segments each road in a mutually exclusive way. The effects of intersecting roads on the segmentation and source point creation is not treated in this very first version of the solution. This issue is therefore considered as an immediate future extension.

Figure 6: Result of implementation with C = 25

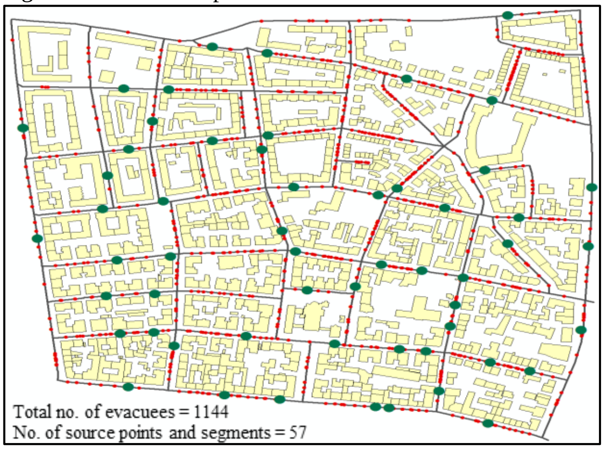
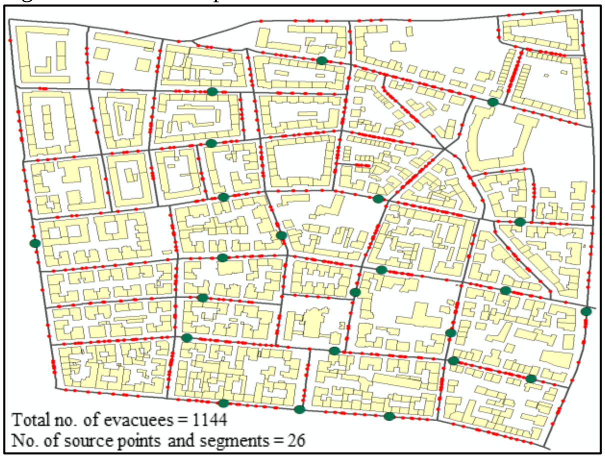


Figure 7: Result of implementation with C = 50



4 Conclusion

We have presented a first solution to extend the bus-based evacuation problem (BBEP) in a way to overcome the restrictions of pre-defined source points by a road segmentation approach which automatically creates these source points. One advantage of this approach is, that the created source points could be used as input for existing BBEP algorithms to optimize the whole evacuation procedure.

The work presented can be seen as a first solution for a very complex problem, which leaves a number of tasks for future work. It is clear that the segmentation strategy is depending on the way how the sequence of roads is treated and also from the accessibility of the roads. The latter has been neglected in this paper and the treatment of the sequence of roads also has to be optimized in future.

References

[1] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[2] Douglas R. Bish. Planning for a bus-based evacuation. *OR spectrum*, 33(3):629–654, 2011.

[3] Horst W. Hamacher and Stevanus A. Tjandra. Mathematical modelling of evacuation problems: a state of the art. In *Pedestrian and Evacuation Dynamics*, pages 227–266. Springer, Berlin, 2001.

[4] Lu, Qingsong, Betsy George, and Shashi Shekhar. Capacity constrained routing algorithms for evacuation planning: A summary of results. In *Advances in spatial and temporal databases*, pages 291-307. Springer Berlin Heidelberg, 2005.

[5] Marc Goerigk and Bob Grün. A robust bus evacuation model with delayed scenario information. *OR Spectrum*, 36(4):923–948, 2014.

[6] Marc Goerigk, Bob Grün, and Philipp Heßler. Branch and bound algorithms for the bus evacuation problem. *Computers & Operations Research*, 40(12):3010–3020, 2013.

[7] Marc Goerigk, Bob Grün, and Philipp Heßler. Combining bus evacuation with location decisions: A branch-and-price approach. *Transportation Research Procedia*, 2(0):783 – 791, 2014. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands.

[8] Marc Goerigk, Kaouthar Deghdak, and Philipp Heßler. A comprehensive evacuation planning model and genetic solution algorithm. *Transportation Research Part E: Logistics and Transportation Review*, 71(0):82 – 97, 2014.

[9] Marina Yusoff, Junaidah Ariffin, and Azlinah Mohamed. Optimization approaches for macroscopic emergency evacuation planning: a survey. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 3, pages 1–7. IEEE, 2008.

[10] Nezih Altay and Walter G Green III. Or/ms research in disaster operations management. *European Journal of Operational Research*, 175(1):475–493, 2006.

[11] Tanka Nath Dhamala. A survey on models and algorithms for discrete evacuation planning network problems. *Journal of Industrial and Management Optimization*, 11(1):265–289, 2015.

[12] Technische Universität Kaiserslautern, Fachbereich Mathematik. Transit Dependent Evacuation Planning For Kathmandu Valley: A Case Study. Report In Wirtschaftsmathematik (WIMA Report). Kaiserslautern: Technische Universität Kaiserslautern, 2014.