# Addressing Congestion in Time-Expanded Networks: A Lifeboat Allocation Model for Maritime Evacuations

BY Andres Velez

DISCUSSION PAPER

Institutt for foretaksøkonomi
Department of Business and Management Science

**FOR 2/2025**
**ISSN: 2387-3000**
January 2025

# Addressing Congestion in Time-Expanded Networks: A Lifeboat Allocation Model for Maritime Evacuations

Andres Velez
Department of Business and Management Science
NHH Norwegian School of Economics, Bergen, Norway
andres.correa@nhh.no ORCID ID 0000-0002-1214-8737

This paper addresses the challenge of congestion in time-expanded networks, focusing on a case study related to maritime evacuations. The problem is made complex by an endogenous relationship between inputs and outputs, where the assignment of flow to an edge leads to increased congestion, which reflects in later arrivals and changes on the overall network topology. This dynamic interaction between flow and congestion is central to the problem, as it results in a feedback loop that complicates the identification of optimal evacuation paths. The study presents an iterative algorithm inspired by the network simplex method, designed to handle the evolving nature of congestion while minimizing evacuation time.

Keywords: Onboard safety, Guidance system, Wireless communication, Lifeboat Assignment, Network flows
January 22, 2025

# Addressing Congestion in Time-Expanded Networks: A Lifeboat Allocation Model for Maritime Evacuations

Andres Velez
Department of Business and Management Science, NHH Norwegian School of Economics, Bergen, Norway.
andres.correa@nhh.no.

This paper addresses the challenge of congestion in time-expanded networks, focusing on a case study related to maritime evacuations. The problem is made complex by an endogenous relationship between inputs and outputs, where the assignment of flow to an edge leads to increased congestion, which reflects in later arrivals and changes on the overall network topology. This dynamic interaction between flow and congestion is central to the problem, as it results in a feedback loop that complicates the identification of optimal evacuation paths. The study presents an iterative algorithm inspired by the network simplex method, designed to handle the evolving nature of congestion while minimizing evacuation time. While the primary case study involves cruise ship evacuations, the approach is generalizable to other scenarios where congestion and non-linear flow dynamics are significant factors. By considering lifeboat capacity, passenger mobility restrictions, and the impact of congestion on network structure, this work provides a practical initial plan for an evacuation off-shore, considering congested, time-expanded network settings.

*Key words*: Congestion, Time-expanded networks, Network simplex method, Congested networks, Evacuation planning

## 1. Introduction

Congestion in network flow problems arises when the demand on a network's resources exceeds its capacity, causing delays, bottlenecks, and inefficiencies. This issue is particularly prevalent in flow problems (for instance, transportation, logistics, and emergency evacuation scenarios), where the flow of people, vehicles, or goods must be carefully managed to avoid undesirable consequences. In such systems, the interaction between network topology, edge capacities, and flow dynamics can cause significant disruptions, particularly when the relationship between the number of users and the speed of movement is non-linear, as observed by Fruin (1970), Fruin (1971), and later, Helbing et al. (2007) and Schadschneider et al. (2009). In these cases, congestion emerges not only as a result of volume, but also from the complex interactions between agents moving through the system and the infrastructure's capacity to handle them.

A fundamental challenge in addressing network flow problems under congested conditions lies in the dynamic evolution of network performance as flow accumulates, complicating the identification of an optimal solution from the outset. In an uncongested environment, flow can typically be allocated based on predetermined costs, enabling the direct application of optimization algorithms to determine shortest paths or maximize throughput. However, under congestion, this assumption no longer holds. While the nominal capacity of each edge remains fixed, increased flow results in higher travel times and diminished effective throughput, constraining the viability of certain routes. As congestion builds within a corridor initially identified as optimal, its usability declines, this introduces an evolving network state where paths that initially appear optimal may degrade as congestion builds. This dynamic interplay introduces substantial complexity, as routes that appear optimal at the outset may become suboptimal as congestion propagates through the network.

The paper by Chalmet et al. (1982) was the pioneering work on building evacuations using a time-expanded network approach, minimizing the evacuation time. After this work was published, several contributions followed the research line such as: Choi et al. (1988) introducing flow-dependant edge capacity constraints, Hamacher and Tufekci (1987) studying how to elude solutions involving cycling inside the building, and Hoppe and Tardos (2000) introducing the first polynomial time algorithm to solve this kind of problems. However, for all these early approaches the network size proved to be a challenge, and limited its applications in real-size planning and operational problems. Thus, heuristic solutions were proposed by Lu et al. (2003), Lu et al. (2005), and Mishra et al. (2018).

In an interesting paper by Kim et al. (2007), the Intelligent Load Reduction (ILR) and the Incremental Data Structure (IDS) algorithms were introduced. Focusing on large-scale evacuations in case of natural or man-made disasters, the ILR algorithm presents the idea of reducing the bottleneck saturation by diverting routes, and in this way, the computation times are reduced and beat benchmarks. However, pedestrian evacuations face challenges such as the consideration of full-mobility and disabled evacuees; in particular, this was studied by Noh et al. (2016) in their paper on the evacuation of a heterogeneous population. Also, we can find that merely considering the bottleneck capacity is often not enough to model the complex interactions of pedestrian flows; since the relationship between the users' density within a corridor and walking speed is not linear as noted previously.

This relationship was considered within the contribution made by Oh et al. (2019); in this work, the curve displaying the interaction between the edge travel time and evacuees were transformed into a piece-wise linear function, and the kinks of such a function were used to define congestion levels. Their network structure incorporates these congestion levels as multiple edge classes, in such a way that a node has several outgoing edges toward the next location arriving at different times. Thus, the constraints make sure that there will only be flow in one of these edges, based on the number of units traversing between two locations. The challenge presented by the congestion affecting the network's topology is addressed by the authors by increasing the network size to provide flexibility, abandoning the networks flows' realm and going for a **MIP** formulation, and finally, they suggested a heuristic algorithm able to cope with the increased computational complexity.

As a case study we explore maritime emergencies since to the best of our knowledge they remain neglected in the OR literature, and the pedestrian dynamics involved in this emergencies is a good example of congested networks. To provide more context, the maritime industry faces significant safety challenges due to the large number of passengers traveling on ships, with fire being a primary concern. Despite efforts to enhance safety measures, incidents continue to occur, as evidenced by 448 significant accidents involving cruise ships reported since 2005 (MIG 2021). The average cruise length in 2021 was 6.6 days, with variations across different markets (CLIA 2021), indicating limited time for passengers to familiarize themselves with the ship's layout. On top of this factors such as the similarity between corridors or lack of real-time information on the hazard's location make navigating ship interiors challenging during emergencies.

Recent developments in wireless communication technology, notably sensor mesh technologies, offer promising solutions for real-time passenger tracking and feedback. They open the possibility of addressing these safety concerns by the implementation of a flexible onboard guidance system to assist passengers in emergencies (Luo 2019).

Drawing from the disaster management principles presented in the literature (Southworth 1991, Tuydes 2005), our previous paper (Velez and Wallace 2025) emphasized the importance of defining destinations for evacuees, and allocating evacuation routes. In the particular case of emergencies on cruise vessels, the first of these two principles can be seen as a **lifeboat assignment**. Despite the consideration of several mechanisms affecting the pedestrian flow (for instance, groups with differentiated mobility restrictions, walking speed, and congestion, among others), finding the relative best assignment cannot be achieved by the mere use of networks without a time dimension.

In this paper, we try a different approach to address the same challenge while keeping the original network size, our approach will make use of network properties and a reinterpretation of some of the simplex algorithm's principles to outline an iterative algorithm. Despite the similarities between building evacuations and off-shore evacuations, the latter problem must take into account the limited capacity of lifeboats, as simply evacuating people from the system is not sufficient. Additionally, the possibility of secondary disasters occurring while adrift on the ocean in lifeboats cannot be overlooked. This adds extra considerations to be taken when solving this particular problem, but we believe our algorithm can be applied to more general problems that consider congested time-expanded networks with a non-linear relationship between edge density and speed.

The remainder of the paper is structured as follows: Section 2 discusses the problem description. Section 3 details the notation. Section 4 presents the network simplex-inspired algorithm for lifeboat assignment. Section 5 presents the technical support and ship geometry. Section 6 provides a case study. Section 7 presents the results of the study. Finally, Section 8 summarizes the findings offering conclusions and an outlook for future research.

## 2. Problem description

The problem at hand involves clearing a congested network in the shortest time possible. The specific case study is the evacuation of a cruise vessel in an emergency situation. The challenge arises from the need to manage the movement of passengers through the vessel's network, where congestion can build up quickly due to limited lifeboat capacities and the varying mobility needs of evacuees.

Each lifeboat has a fixed capacity, and it is critical to ensure that passengers with mobility restrictions are distributed in a way that ideally their proportion does not exceed a predefined threshold. This constraint introduces a trade-off between minimizing the overall evacuation time, the limited seats in each lifeboat and maintaining an appropriate balance of evacuees, particularly those with special needs.

This problem is formally represented in Appendix A. Given the computational complexity of directly solving this problem, our approach focuses on developing an algorithm that can efficiently manage these constraints and minimize evacuation time in a practical manner. Before explaining the core concepts of our contributions, we will shortly explain the input data and considerations necessary for its implementation.

### 2.1. Network representation

The time-expanded network representation proposed by Ford and Fulkerson (1958) remains foundational for modeling dynamic systems, treating time as a finite and discrete sequence of steps. In this study, the cruise vessel layout is modeled as a network expanded over time, where nodes are spaced at consistent intervals. Ensuring uniform edge lengths between adjacent nodes in the static network is essential for maintaining a fixed time step size, enabling travel times between connected nodes to be represented as integer values.

To account for heterogeneous mobility restrictions, this model incorporates differences in walking speeds among passenger groups, a critical factor in determining traversal times. Following the guidelines in Wang et al. (2021), we consider a maximum average walking speed of 2 m/s for young adults and 1.5 m/s for elders. This leads us to select a time step size that ensures travel times between the same locations are integer values for both walking speeds simultaneously. Such a configuration is feasible for walking speeds with a difference of approximately 25%. For the walking speeds considered, an edge length of 6 meters with a uniform time step of 1 second satisfies the criteria, ensuring that traversal times between connected nodes remain consistent and integer-valued across all passenger groups.

Variations in passenger mobility levels shape the network structure by introducing multiple flow types, or commodities, within the network, as discussed in Velez and Wallace (2025). This distinction is particularly significant in configuring the network, where one possible approach involves designating certain edges exclusively for specific groups. Alternatively, a layered mode-space-time network representation, inspired by SteadieSeifi et al. (2017) in the context of perishable goods transportation, can be employed. In this representation, the space-time mapping is duplicated for each passenger group, creating distinct layers. Each layer consists of nodes connected by edges that reflect traversal times at the walking speed of the corresponding group (see Fig. 1). To address the shared physical space and interactions between groups, connections are established between these layers, forming an integrated network.

Since congestion is directly influenced by the interplay between both groups of people and the capacity of every corridor to accommodate individuals, it cannot be accounted for each layer individually. Unlike conventional network theory, where capacity is typically computed on edges, the layers under consideration do not share the same edges that connect the nodes. To overcome this challenge, we suggest calculating capacity over nodes, as they remain consistent across all layers and serve as the crucial link between layers, addressing the interaction between different commodities.

### 2.2. Calculating the congestion

In addition to the expanded network size and the subsequent computational complexity, incorporating congestion effects in such networks presents significant challenges. Temporal connections between locations are influenced by passengers' walking speeds, where faster individuals traverse distances in fewer time steps. However, when a large group moves simultaneously toward a destination, individual walking speeds diminish due to crowding, resulting in more time steps to cover the same distance. Consequently, the solution structure directly impacts the network topology.

To estimate corridor occupancy accurately, one might consider aggregating passenger counts at each node across solution paths. However, this approach often underestimates congestion levels. For instance, in Figure 1, suppose group 1 arrives at $(node_1, t_2)$; if another individual departs from $(node_1, t_1)$, we know they have yet to reach $(node_2, t_6)$, increasing crowding at the original node. Yet, simply considering head and tail nodes of each path does not capture this, as flow is not continuous. To address this, we implemented "holding edges," illustrated in Figure 1. These

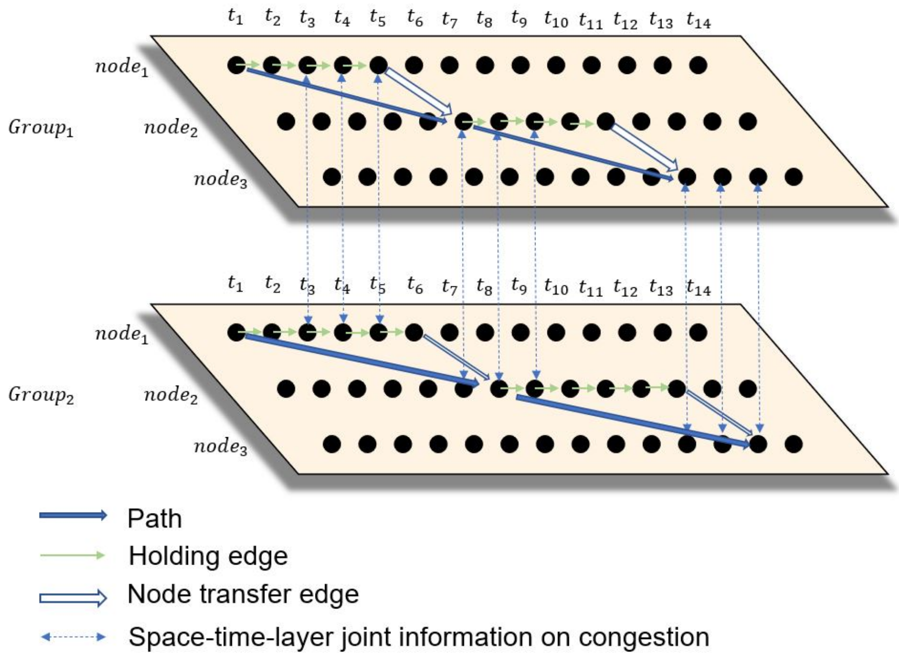


**Figure 1** **Example of a space-time-layer representation of the network. The same physical path is represented for both groups but the arrival time is delayed from the group with the slower walking speed.**

edges ensure that transfers between nodes occur only at the final time step, thereby preserving temporal continuity; the flow is assigned to each holding edge accordingly. This approach enables precise tracking of passenger positions throughout the network.

## 3. Notation

Let $G = (V, E)$ be a time-expanded network, where $V$ represents its node set and $E \subset \{(u, v) \mid u, v \in V, u \neq v\}$ its edge set. Each edge $(u, v) \in E$ has an associated time cost $t_{uv}$ that matches the gap in the time-axis between both nodes. Each node $v \in V$ has a congestion level $\rho_v$, which represents the maximal occupancy at that node at any given time. Let $P$ be the current path from the source node $s$ to the destination node $d$, and $P'$ an alternative path starting from a crossroad node $v_j$ to $d$. The total time cost of a path $P$ is denoted by $C(P)$. Finally, let $T_c$ represent the clearing time, defined as the maximum arrival time across all paths in the network.

## 4. Network Simplex-Inspired algorithm for lifeboat assignment

To the best of our knowledge, there has been limited research on the application of congested-time-expanded networks. In contrast to the work of Oh et al. (2019), which addressed this gap by introducing distinct edges corresponding to different congestion levels, this paper adopts an iterative approach, outlined at a high level in Algorithm 1 in Appendix B. The proposed algorithm consists of two primary mechanisms, which will be discussed in detail in the following subsections.

Additionally, our approach requires an initial solution that will be improved and serves to determine the initial network congestion.

In their work, Velez and Wallace (2025) introduced a lifeboat assignment model designed for static networks, combining two optimization problems. First, a flow profile is found in a multi-commodity flow problem, followed by solving a flow decomposition problem responsible for defining the origin-destination pairs. Since the formulation in the paper is purely static, the authors also highlight the potential overestimation of congestion within corridors in their method. Without temporal information, it is only possible to estimate the popularity of an edge, not the real network condition. If the passengers walking through a seemingly congested corridor in the static network are in reality evenly distributed over time, it becomes feasible to assign more flow through it, this potentially can even suggest a shorter evacuation time. The solutions derived from the methodology proposed by Velez and Wallace (2025) could serve as a good initial solution for a time-expanded formulation, due to how fast it is possible to find a good enough solution in a static network.

The proposed algorithm incorporates two primary mechanisms: the **Network Topology Update Mechanism** and the **Improving Mechanism**. These mechanisms work iteratively to balance the congestion in the network. The Improving Mechanism focuses on optimizing the latest arrival time by refining the flow distribution across the network. If no further improvements can be made based on the current network state, the Network Topology Update Mechanism is triggered. This mechanism modifies the network topology by adjusting the edges based on the current flow conditions. A new network configuration is then discovered, and the algorithm attempts to improve the solution again. This iterative process continues until the solution and network congestion are balanced. In some cases, the algorithm may revisit previously encountered solutions, necessitating the implementation of an escape procedure to prevent infinite loops.

### 4.1. Network topology update mechanism

This mechanism integrates the impact of congestion within time-expanded networks, enabling a systematic evaluation of modifications applied in previous iterations. After each iteration, edges connecting space-time nodes are reassessed based on updated congestion levels. If a node $A$ exhibits significant congestion at a specific time step relative to its previous state, its outgoing edges, along with those from selected time copies, are removed and replaced with alternatives that lead to the destination at later time step. For this study, outgoing edges from four consecutive time copies of the congested node are removed, adjusted to the longest traversal time from a node in a non-congested network.

Conversely, when congestion levels decrease, the network can incorporate edges representing earlier arrival times. These additions are made at the end of an iteration, introducing opportunities to reduce path lengths by alleviating congestion along critical nodes (See Fig. 2). This dynamic adjustment mechanism serves as a central strategy for optimizing evacuation times in the presence of evolving congestion patterns.

Congestion affects walking speeds in a nonlinear manner, with increasing congestion leading to slower movement. Empirical findings from Helbing et al. (2007) identify behavioral shifts in average flow at local densities of approximately $3.5\,\mathrm{m}^{-2}$ and $7\,\mathrm{m}^{-2}$. Based on these results, we define congestion thresholds $\rho_{\text{threshold}}$ that divide congestion into three levels: mild congestion ($0 \leq \rho < 3.5\,\mathrm{m}^{-2}$), moderate congestion ($3.5 \leq \rho < 7\,\mathrm{m}^{-2}$), and heavy congestion ($\rho \geq 7\,\mathrm{m}^{-2}$). These congestion intervals influence the effective lengths of edges between physical spaces. Updates to the network's layers preserve a first-in-first-out (FIFO) condition, ensuring that passengers are routed in accordance with their original order of arrival per group.

### 4.2. Improvement Mechanism

This mechanism serves as the core of the system, responsible for identifying new paths and reallocating passengers from the currently longest path (refer to Algorithm 2 for detailed implementation). Below, we outline the tasks within this algorithm and describe each procedural step.

**4.2.1. Alternative Path Construction** In the network-simplex algorithm, identifying negative cost cycles is central to discovering opportunities for improvement. Analogously, our approach identifies alternative paths that offer quicker arrival times or reduced congestion, thereby enabling the reallocation of passengers in a manner that improves the overall clearing time. The following steps outline the detailed procedure for constructing these alternative paths:

1. **Path Splitting**

Let

$$P = (s, v_1, v_2, \ldots, v_k, d) \tag{1}$$

denote a path taken by a passenger from the starting node $s$ to the destination $d$, which represents a lifeboat. To address congestion along this path, we first identify the most congested node

$$v_c \in P,$$

where congestion is defined by the maximal level of congestion $\rho_{v_i}$ at that node. Specifically, $v_c$ satisfies

$$\rho_{v_c} = \max_{v_j \in P} \rho_{v_j},$$

ensuring that the congestion at $v_c$ is greater than or equal to the congestion of every other node along $P$.

2. **Crossroad Selection**

Next, the algorithm selects the starting node $s$ or the earliest crossroad node $v_j$ on $P$ that precedes the congested node $v_c$ to serve as the origin for a new sub-path. Note that the selection of the earliest crossroad does not imply that the detour is taken there, the new sub-path can follow $P$ until it diverts on a later crossroad, if the arrival time is improved.

3. **Alternative Path Identification**

We then construct a new path

$$P' = (v_j, u_1, u_2, \dots, d) \tag{2}$$

from the crossroad $v_j$ to the destination $d$, avoiding the congested node $v_c$ and its time copies by finding the quickest possible arrival to the lifeboat of interest. The cost of path $P'$ is defined as the sum of the travel times along all edges of $P'$:

$$C(P') = \sum_{(u,v)\in P'} t_{uv}, \tag{3}$$

where $t_{uv}$ is the travel time for each edge. If

$$C(P') < C(P) \tag{4}$$

then some or all passengers from $P$ are reassigned to the new path $P'$. Note that selecting $P'$ such as its costs equals $P$'s cost might also leads to reducing congestion, this could happen if we can remove enough passengers from the longest path to relax the congestion in one of its nodes, but the number is passengers is not enough to increase the congestion in any node of the shorter path to the next level.

#### 4.2.2. Shortest Path in a Directed Acyclic Graph (DAG):

**Theorem 1** *The shortest path problem in a time-expanded network represented as a Directed Acyclic Graph (DAG) can be solved in* $\mathcal{O}(|V| + |E|)$ *time using topological ordering.*

*Proof.* In a DAG $G = (V, E)$, nodes can be topologically ordered so that for every edge $(u, v) \in E$, $u$ precedes $v$. This ordering enables processing each node only once, allowing us to initialize labels to infinity (except the source, set to zero) and update labels in a single pass. For each node $u$, we relax its outgoing edges $(u, v)$ by setting $\text{label}(v) = \min(\text{label}(v), \text{label}(u) + w(u, v))$. Since each edge is relaxed exactly once, the time complexity is $\mathcal{O}(|V| + |E|)$ and it represents the best possible time complexity for any shortest path algorithm Ahuja et al. (1993).■

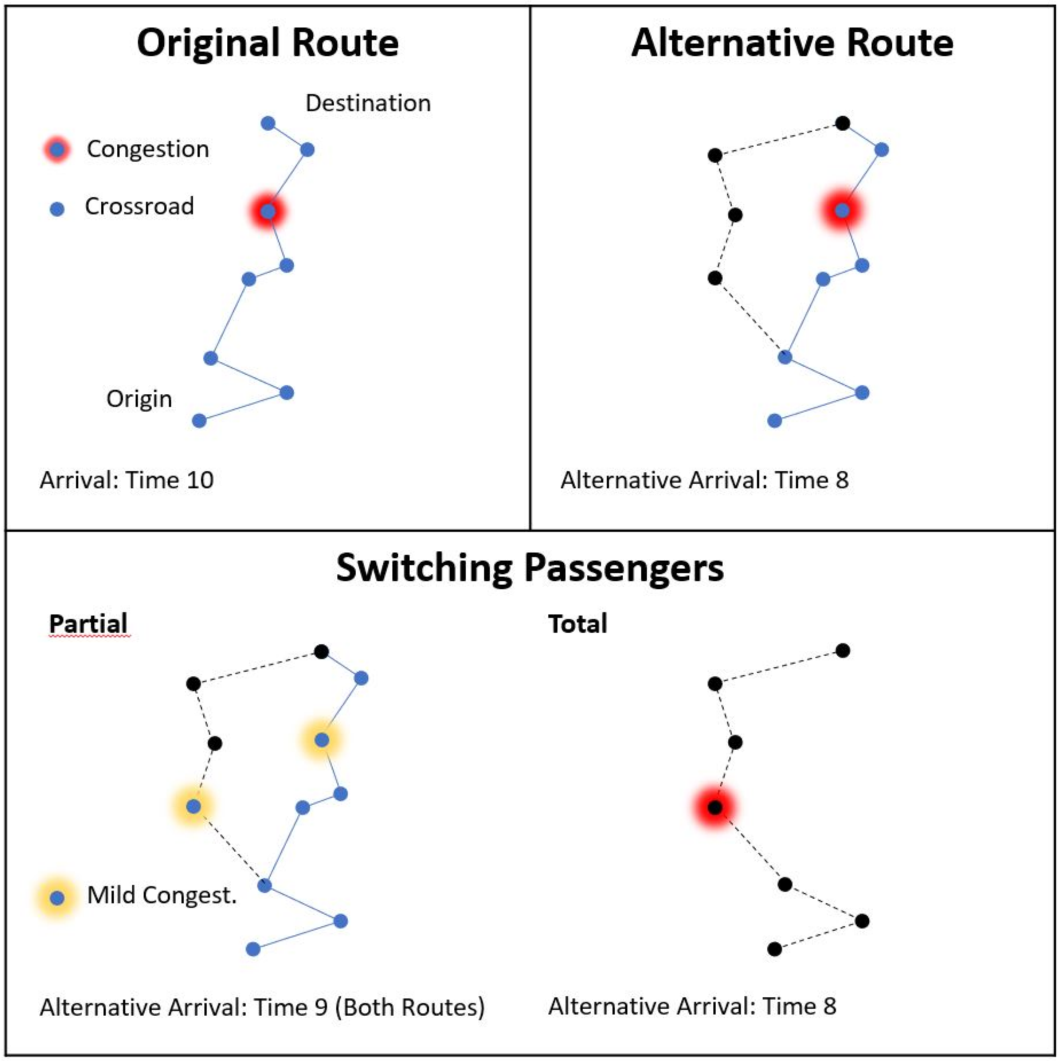


**Figure 2** **An example of constructing directed cycles in a time-expanded network. The top left image depicts an original route with a traversal time of 10 steps, highlighting the most congested node (in red). The top right image introduces an alternative route that detours from a preceding node. The bottom image illustrates two possible outcomes when redistributing passengers: a partial transfer reduces congestion and traversal time on the original route, while a full transfer eliminates it from the solution.**

In our approach, the earliest feasible arrival at a specific destination is computed by examining a time-expanded network, which is analyzed using the shortest path labeling-correcting algorithm outline in the book by Ahuja et al. (1993). Within this network, each node is represented by multiple time-stamped copies, each corresponding to a distinct point in time. The algorithm iteratively updates the shortest path values associated with these nodes, propagating valid (finite) values throughout the network. Upon completion of the computation, the first time-stamped node with a valid shortest path value is identified, representing the earliest possible arrival time at the destination.

**4.2.3. Determining the Flow to Exchange** The goal of this step is to determine the flow $f$ to be transferred from the congested path $P$ to the alternative path $P'$ to reduce the clearing time $T_c$ while alleviating congestion.

**Lemma 1:** *For a given alternative path $P'$, transferring flow $f$ from the congested path $P$ to $P'$ reduces the clearing time $T_c$ by either relaxing congestion on $P$ or fully reallocating its flow to $P'$.*

*Proof.* Each node $v_i$ in the path $P = (s, v_1, v_2, \ldots, v_k, d)$ has a congestion level $\rho_{v_i}$, and congestion thresholds $\rho_{\text{threshold}}$ dictate levels where traversal times increase. To minimize $T_c$, the algorithm computes $f$ as the smallest flow adjustment needed to relax congestion on $P$ or maintain steady conditions on $P'$:

$$f = \min\left(\Delta\rho_{v_i}, \Delta\rho'_{v_i}\right),$$

where $\Delta\rho_{v_i}$ is the flow reduction needed for $\rho_{v_i}$ to fall below its threshold, and $\Delta\rho'_{v_i}$ is the additional flow $P'$ can handle before reaching its next threshold. If the flow on $P$ is fully transferred to $P'$, $P$ is removed from the solution. In the opposite case, to attempt removing $P$, a new alternative path $P'$ is selected starting from the second quickest arrival to the destination node, and the process continues accordingly. ■

However note that congestion on shared edges can persist even after removing $P$ from the solution, as flows from other paths may still contribute to high $\rho$ values.

**4.2.4. Variable Neighborhood Descent (VND):** The improving mechanisms previously described can be applied under different goals, enabling exploration of the solution space. The Directed Acyclic Graph (DAG) shortest path algorithm can be utilized to identify solutions either within the lifeboat assigned to the last passenger or in other lifeboats. Drawing inspiration from the **Variable Neighborhood Descent** heuristic, a sequence of neighborhoods is defined, with the algorithm restarting its search from the first neighborhood upon finding a new best solution. It is important to notice that the construction of the neighborhood was driven by the evacuation's objectives of our study case, for other problems in congested time-expanded networks the search space exploration must be addressed following their particular needs.

The improving algorithm, detailed in Appendix B, applies the neighborhoods in the following order. First, alternative solutions are explored within the originally assigned lifeboat. Next, options are examined in lifeboats with available capacity. Finally, for lifeboats at full capacity, solutions are sought by swapping passengers from the same group (namely young and elders) between lifeboats.

**4.2.5. Stopping Criteria:** Two stopping criteria are implemented to ensure the algorithm terminates effectively. First, considering the need for a rapid response during emergencies, a maximum runtime can be imposed on the improving mechanism. This ensures that the network update mechanism can execute within the time constraints of the application.

Second, a degenerate case arises when the algorithm oscillates between two solutions, swapping flow indefinitely. Suppose two paths, $P_1$ and $P_2$, have arrival times $t_n + 1$ and $t_n$, respectively. In one iteration, flow is transferred from $P_1$ to $P_2$, reducing congestion on $P_1$ and setting its arrival time to $t_n$, while increasing congestion on $P_2$, resulting in $t_n + 1$. In the next iteration, the process reverses, returning the algorithm to its original state. In such cases, the algorithm must detect the loop and return one of the two solutions, accepting $t_n + 1$ as the best achievable arrival time.

# 5. Technical Support and Ship Geometry

## 5.1. Ship Geometry and evacuation simulation

The geometry data used in this study were obtained from different works conducted under the EU framework 7 project SAFEGUARD (Safeguard 2012), which carried out a comprehensive analysis of ship evacuation procedures. As part of this project, a semi-unannounced full-scale assembly was conducted on a vessel operated by Royal Caribbean Cruise Lines International, a ship that operates multiple vacation trips in the Caribbean and Baltic Sea regions. The vessel consists of thirteen decks, including seven decks for passenger cabins and additional decks dedicated to entertainment and recreational facilities, such as restaurants, bars, swimming pools, a casino, a theater, a cinema, a spa, a business center, a gym, a climbing wall, a mini-golf course, a card room, and shops (Galea et al. 2013).

To ensure the accuracy and reliability of evacuation models, the SAFEGUARD project answers the invitation from The International Maritime Organization (IMO) for sharing full-scale data to be used for validation and calibration. The methodologies utilized for the data collection for the data sets were clearly depicted in the work by Deere et al. (2012), and experiments related to the response time of the passengers to the evacuation alarm were run by Brown et al. (2021).

The validation data sets, including the ship's geometries, can be accessed at the Fire Safety Engineering Group (FSEG) associated with Greenwich University through the following link: `https://fseg.gre.ac.uk/validation/ship_evacuation`. By utilizing the geometry data derived from the Royal Caribbean cruise vessel described in the aforementioned references, this study ensures the incorporation of validated and reliable information. The obtained geometry data will be appropriately transformed into a suitable data structure to be utilized in the optimization models formulated in this research.

## 6. Case Study

We based our model on the layout of a Royal Caribbean International cruise ship, which accommodates 2,500 passengers and 842 crew members. The vessel is equipped with 18 lifeboats, each with a capacity of 150 passengers, distributed evenly on both sides of the ship. To streamline the model and reduce the number of decision variables, we used the seven possible lifeboat exits as the final destinations, ensuring that no critical information was lost.

The network layers were constructed following two key rules: each intersection became a node, and nodes were also placed every 6 or 12 meters to accurately reflect possible u-turns, which allowed us to find a standardized time step length of 1 second, meeting the integrality requirement outlined in Section 2. The time horizon for the time expansion was set at 250 time steps, resulting in two networks—one for each passenger group. The elders' network comprised 119,250 nodes and 274,108 edges, while the youngsters' network also had 119,250 nodes but with 275,956 edges. This approach allowed us to maintain consistent node counts across layers, enabling the measurement of congestion at nodes, while edge counts varied between groups.

For this paper, we have assumed that all passengers will begin the evacuation process immediately upon hearing the alarm. Although studies exist on the reaction time of passengers to evacuation cues, our focus is on the algorithmic aspects of the problem. As such, certain assumptions have been made to simplify the model and emphasize the computational methods.

Similarly as in our former paper for our numerical tests, we considered two groups of passengers and three different scenarios: (1) all passengers are in their cabins during a nighttime emergency, (2) most passengers are on the top decks during the day, and (3) passengers are in restaurants or near the theater at dinnertime. The largest restaurants and the theater are near the lifeboats, aligning with assembly stations in previous studies.

## 7. Results

For each of the three cases outlined in the former section, our Network Simplex-inspired algorithm was applied to the lifeboat assignment problem. The results are presented for various initial solutions, which were generated using the multi-commodity flow and decomposition approach described in Velez and Wallace (2025), with modifications to the penalty values for congestion. The selected penalty values—1, 2, 10, 20, 30, and 100—reflect different levels of aversion to exceeding corridor capacity, where a value of 1 represents the least restrictive scenario and 100 the most restrictive. Consequently, initial solutions corresponding to higher penalty values are expected to distribute passengers more evenly across the vessel compared to those with lower penalties. All initial solutions were obtained using the Python API of Gurobi.

This study focuses on the algorithmic procedure for addressing problems in congested time-expanded networks. Accordingly, we directly utilize the age distributions obtained from solving the lifeboat assignment problem within a static network. For real-world implementation, it is essential to consider the trade-off between minimizing the network's clearance time and ensuring resilience against potential secondary disasters offshore. This resilience is achieved by maintaining an adequate number of physically capable individuals in each lifeboat to assist in emergencies. Although this aspect falls outside the scope of the present paper, one possible approach involves designing a heuristic method that identifies lifeboats exceeding the predefined threshold for a desired age distribution and subsequently seeks opportunities to reduce the average evacuation time by reallocating passengers across different age groups.

Table 1 presents, for each age group, the percentage of improvement relative to the initial solution provided as input. The values in parentheses indicate the time step at which the last passenger in each group reaches their assigned lifeboat, with the largest of these values representing the network's clear time.

In Case 1, the collected data indicates that the shortest clear time is achieved when no penalties are applied, suggesting that significant bottlenecks are unlikely when passengers are evenly distributed across the decks. Consequently, imposing high congestion penalties results in an extended clear time for the system. This finding implies that the solution obtained through our static approach could be highly effective in such scenarios, particularly given the speed with which assignments can be determined. While there is potential for minor refinements in the allocation of younger passengers without negatively affecting outcomes for older passengers.

In Case 2, the optimal clear time is obtained with a penalty value of 20, which aligns with the recommendation from our previous study. Since passengers are concentrated on the top decks, congestion plays a more critical role compared to Case 1. While the overall improvement in clear time is modest, the best solutions exhibit enhancements of up to 17.36% for the younger age group. The runtime is notably longer than in the other two cases, suggesting increased complexity in balancing congestion and clear time, particularly when initial solutions came from formulations with high penalties. This observation indicates that excessive congestion penalties can disproportionately impact certain passenger groups, particularly younger passengers, by directing them through significantly longer paths. As a result, the algorithm encounters greater difficulty in converging to a solution that improves assignments for younger passengers without compromising outcomes for older passengers.

For Case 3, our previous study identified challenges where certain penalty values led to infeasible formulations, leading to the conclusion that allowing passengers to follow their shortest paths is preferable to actively avoiding congestion. In the present study, high penalty values once again underperformed, reinforcing our prior observation as a viable strategy.

**Table 1 Computational results for different cases at various starting solutions.**

| | | Case 1 | Case 2 | Case 3 |
|---|---|---|---|---|
| 1 | Young | -2,94% (99) | 0% (121) | -8,42% (87) |
| | Elder | 0% (108) | 1,39% (146) | -6,9% (108) |
| | RunningT | 20,84 sec. | 79,52 sec. | 33,3 sec. |
| 2 | Young | -5,88% (102) | -3,31% (117) | -8,42% (87) |
| | Elder | 0% (128) | 2,78% (148) | -6,90% (108) |
| | RunningT | 15,91 sec. | 51,43 sec. | 32,71 sec. |
| 10 | Young | -10,19% (97) | -3,08% (126) | -7,37% (88) |
| | Elder | 0% (108) | -2,70% (144) | 0% (120) |
| | RunningT | 21,48 sec. | 50,66 sec. | 79,75 sec. |
| 20 | Young | -10,81% (99) | -12,59% (118) | -20,00% (92) |
| | Elder | 0% (128) | -4,0% (144) | 0% (117) |
| | RunningT | 27,696 sec. | 52,77 sec. | 16,43 sec. |
| 30 | Young | -28,03% (95) | -17,36% (119) | -0,89% (111) |
| | Elder | -3,12% (124) | -0,67% (148) | 0% (117) |
| | RunningT | 69,6 sec. | 116,04 sec. | 22,54 sec. |
| 100 | Young | -27,66% (102) | -17,36% (119) | 1,79% (114) |
| | Elder | 0% (128) | -0,67% (148) | 0% (117) |
| | RunningT | 33,626 sec. | 113,53 sec. | 44,4 sec. |

**Notes:** The percentages represent the change relative to the initial solution obtained using the static network approach. Negative values indicate a reduction in the network's clear time, reflecting an improvement in evacuation efficiency. Values in parentheses indicate the latest arrival time of any member of the given group. Running time is reported in seconds.

In Appendix A, we present a mathematical formulation of the assignment problem based on Velez and Wallace (2025), extending the static formulation proposed in that work to incorporate time-expanded networks. While an in-depth analysis of this model falls beyond the scope of this paper, we consider it valuable to report the results summarized in Table 2. This table displays the outcomes obtained using the Gurobi Solver for a single iteration, noting that the solutions to our problem ultimately determines the network's topology, requiring again an iterative process to address its endogeneity.

Additionally, the first row of the table reports results from solving a relaxed version of the problem without congestion constraints. This allows us to observe that congestion constraints significantly increase the complexity of the problem, suggesting that a more effective approach might involve math-heuristic methods. However, the overall feasibility of an exact approach appears limited. Even in the absence of complicating constraints, obtaining a solution for the first iteration requires 60 seconds. Given the real-time nature of our application, which demands immediate responsiveness, a fully balanced solution would likely require computational times far exceeding practical limits. In this context, our network simplex-inspired approach emerges as a promising alternative for applications requiring rapid decision-making.

**Table 2** **Computational results for the mathematical formulation approach for one iteration.**

| Penalty | Case | Clear Time | Gap | Runtime(sec) |
|---|---|---|---|---|
| NC | 1 | 108 | 0 % | 65,67 |
| | 2 | 128 | 0 % | 61,94 |
| | 3 | 96 | 0 % | 2,69 |
| 2 | 1 | 132 | 2,17 % | 23372 |
| | 2 | 168 | 14,71 % | 72893 |
| | 3 | 196 | 3,00 % | 85,73 |
| 30 | 1 | 160 | 4,71 % | 10793 |
| | 2 | 172 | 2,00 % | 24905 |
| | 3 | 192 | 0,20 % | 80,66 |
| 100 | 1 | 204 | 1,92 % | 9260 |
| | 2 | 204 | 3,22 % | 2972 |
| | 3 | 136 | 0,01 % | 69,22 |

## 8. Conclusion

Modeling congestion in networks poses substantial computational challenges due to its dependence on the solution structure, the resulting increase in problem size, and the difficulty of accurately capturing its effects. Despite these complexities, accounting for congestion is essential to prevent misleading solutions. This study presents a methodological framework that balances computational efficiency with a sufficiently detailed congestion representation, ensuring high solution quality while maintaining tractability.

The methodology developed in this work is designed to generate solutions rapidly. By formulating the problem within the well-established domain of Network Flow Problems, we leverage known properties related to computational complexity and performance. However, this choice also imposes constraints on congestion modeling, as traditional network flow formulations struggle to represent dynamic congestion effects with high accuracy.

To address this, we implement a discretized congestion model in which predefined congestion levels introduce fixed delays within the network. While widely adopted in the literature and computationally efficient, this approach entails an inherent trade-off: congestion effects are represented as stepwise increments rather than continuous functions of crowd density. Consequently, extreme congestion scenarios—where pedestrian movement is severely restricted beyond the highest predefined congestion level—may be underestimated.

Although our results indicate that this methodology provides a robust foundation for a reactive guidance system, as described in our previous work, further refinements in congestion modeling could enhance solution quality while preserving computational efficiency. A natural extension of this study involves exploring continuous congestion models that remain compatible with the network flow formulation. Such an approach would enable a more precise representation of passenger movement while retaining the key advantage of generating an initial near-optimal solution for real-time adjustments.

## Compliance with Ethical Standards

**Conflict of interest** The authors have no competing interests to declare that are relevant to the content of this article.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Funding** No funds, grants, or other support was received. The authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

## References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice hall.

Brown, R., Galea, E., Deere, S., Filippidis, L., 2021. Passenger response time data- sets for large passenger ferries and cruise ships derived from sea trials. International Journal of Maritime Engineering 155. doi:10.5750/ijme.v155iA2.899.

Chalmet, L.G., Francis, R.L., Saunders, P.B., 1982. Network models for building evacuation. Management Science 28, 86–105. URL: http://www.jstor.org/stable/2631076.

Choi, W., Hamacher, H., Tufekci, S., 1988. Modeling of building evacuation problems by network flows with side constraints. European Journal of Operational Research 35, 98–110. URL: https://www.sciencedirect.com/science/article/pii/0377221788903827, doi:https://doi.org/10.1016/0377-2217(88)90382-7.

CLIA, 2021. Overview of the global cruise industry. https://cruising.org/-/media/clia-media/research/2022/2021-1r-clia-001-overview-global.ashx.

Deere, S., Galea, E., Filippidis, L., Brown, R., 2012. Data collection methodologies used in the safeguard project to collect human factors data, pp. 13–24. doi:10.3940/rina.safe.2012.02.

Ford, L.R., Fulkerson, D.R., 1958. Constructing maximal dynamic flows from static flows. Operations Research 6, 419–433. URL: https://doi.org/10.1287/opre.6.3.419, doi:10.1287/opre.6.3.419, arXiv:https://doi.org/10.1287/opre.6.3.419.

Fruin, J.J., 1970. Designing for pedestrians a level of service concept. Polytechnic University.

Fruin, J.J., 1971. Pedestrian planning and design. Technical Report.

Galea, E., Deere, S., Brown, R., Filippidis, L., 2013. An experimental validation of an evacuation model using data sets generated from two large passenger ships. Journal of Ship Research 57, 155--170. doi:10.5957/JOSR.57.3.120037.

Hamacher, H.W., Tufekci, S., 1987. On the use of lexicographic min cost flows in evacuation modeling. Naval Research Logistics (NRL) 34, 487--503. URL: https://EconPapers.repec.org/RePEc:wly:navres:v:34:y:1987:i:4:p:487-503.

Helbing, D., Johansson, A., Al-Abideen, H.Z., 2007. Dynamics of crowd disasters: An empirical study. Physical Review E 75. doi:10.1103/physreve.75.046109.

Hoppe, B., Tardos, E., 2000. The quickest transshipment problem. Mathematics of Operations Research 25, 36--62. URL: http://www.jstor.org/stable/3690422.

Kim, S., George, B., Shekhar, S., 2007. Evacuation route planning: scalable heuristics, in: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems, pp. 1--8.

Lu, Q., George, B., Shekhar, S., 2005. Capacity constrained routing algorithms for evacuation planning: A summary of results, in: International symposium on spatial and temporal databases, Springer. pp. 291--307.

Lu, Q., Huang, Y., Shekhar, S., 2003. Evacuation planning: a capacity constrained routing approach, in: International Conference on Intelligence and Security Informatics, Springer. pp. 111--125.

Luo, M., 2019. How to Guide Emergency Evacuations on Cruise Ships? Modelling with Optimization and Simulation Methodology. Master's thesis. Norwegian School of Economics.

MIG, 2021. Cruise ship safety statistics. https://www.maritimeinjuryguide.org/blog/cruise-ship-safety-statistics/.

Mishra, G., Mazumdar, S., Pal, A., 2018. Improved algorithms for the evacuation route planning problem. Journal of Combinatorial Optimization 36, 280--306.

Noh, D.j., Koo, J., Kim, B.I., 2016. An efficient partially dedicated strategy for evacuation of a heterogeneous population. Simulation Modelling Practice and Theory 62, 157--165.

Oh, C.H., Kim, M.H., Kim, B.I., Ko, Y.M., 2019. An efficient building evacuation algorithm in congested networks. IEEE Access 7, 169480--169494. doi:10.1109/ACCESS.2019.2955477.

Safeguard, 2012. Overview of the global cruise industry. https://cordis.europa.eu/project/id/218493.

Schadschneider, A., Klüpfel, H., Kretz, T., Rogsch, C., Seyfried, A., 2009. Fundamentals of Pedestrian and Evacuation Dynamics. pp. 124--154. doi:10.4018/9781605662268.ch006.

Southworth, F., 1991. Regional evacuation modeling: a state-of-the-art review. Technical Report. Oak Ridge National Laboratory.

SteadieSeifi, M., Dellaert, N., Nuijten, W., Van Woensel, T., 2017. A metaheuristic for the multimodal network flow problem with product quality preservation and empty repositioning. Transportation Research Part B: Methodological 106, 321--344. URL: https://www.sciencedirect.com/science/article/pii/S0191261516304039, doi:https://doi.org/10.1016/j.trb.2017.07.007.

Tuydes, H., 2005. Network traffic management under disaster conditions. .

Velez, A., Wallace, S.W., 2025. Active guidance systems in maritime evacuations: optimizing the lifeboat allocation for cruise ships. Annals of Operations Research URL: https://doi.org/10.1007/s10479-025-06467-1, doi:10.1007/s10479-025-06467-1.

Wang, X., Liu, Z., Wang, J., Loughney, S., Yang, Z., Gao, X., 2021. Experimental study on individual walking speed during emergency evacuation with the influence of ship motion. Physica A: Statistical Mechanics and its Applications 562, 125369. URL: https://www.sciencedirect.com/science/article/pii/S0378437120307202, doi:https://doi.org/10.1016/j.physa.2020.125369.

# Appendix. A

**Table 3 Sets in the proposed model**

| Symbol | Description |
|---|---|
| G | $G = (\mathcal{V}, \mathcal{A}, \mathcal{L})$, expanded graph representing the ship layout in discrete time steps. |
| $\mathcal{V}$ | Collection of vertices, representing a logical space at different times (e.g. a cabin, a hall, a doorway, and an intersection of corridors) and the super-sink node. |
| $\Omega$ | Collection of sink-nodes (e.g. safe boats), $\Omega \subseteq \mathcal{V}$, all of these nodes are connected to a super-sink. |
| $\mathcal{Q}$ | Collection of nodes in which the transit is not allowed. |
| $\mathcal{A}$ | Collection of edges between nodes (e.g. a corridor and a stairway), which can also be expressed as $(i,j)$, where $(i,j) \in \mathcal{V}$. |
| $\mathcal{K}$ | Collection of the groups of evacuees labeled according to their age and mobility restrictions (Elder and Wheelchair users). |
| $\mathcal{T}$ | Collection of considered time intervals. |
| $\mathcal{H}$ | Collection of origin-nodes, $\mathcal{H} \subseteq \mathcal{V}$. |
| $\mathcal{L}$ | Collection of layers in the network G. Each layer collects the traverse time for one or more groups in K. |
| $\mathcal{W}_t$ | Collection of all the paths that finish at time $t \in \mathcal{T}$. |
| $\mathcal{P}_i^l$ | Collection of paths for the node $i \in \mathcal{V} \setminus \{V+1\}$ , $l \in \mathcal{L}$. |
| $\omega_j^l$ | Collection of all paths with ending node $i \in \Omega$ at anytime $t$ in layer $l \in \mathcal{L}$. |
| $\Delta 1_{i,t}$ | Set of all paths $n \in \mathcal{P}^{l1}$ that traverse node $i \in \mathcal{V}$ at time $t$. |
| $\Delta 2_{i,t}$ | Set of all paths $n \in \mathcal{P}^{l2}$ that traverse node $i \in \mathcal{V}$ at time $t$. |

**Table 4 Variables and Parameters**

| Symbol | Description |
|---|---|
| Variables | |
| $\zeta$ | Estimates the earliest clearing time of the network. |
| $x_{i,n}$ | Number evacuees from source $i \in \mathcal{H}$ traversing the path $n \in \mathcal{P}_i^{l1}$. |
| $z_{i,n}^k$ | Number evacuees from source $i \in \mathcal{H}$ and commodity $k \in \mathcal{K}$ traversing the path $n \in \mathcal{P}_i^{l2}$. |
| $y_t^l$ | Binary variable accounting whether the period t is the finishing time of any used path in layer l. |
| $f_{i,t}$ | Total excess of flow in the node $i \in \mathcal{V}$ at time $t \in \mathcal{T}$. |
| $\varphi_j^k$ | Passengers exceeding the recommended age distribution in node $j \in \Omega$ for commodity k. |
| Parameters | |
| $A_i$ | Area of influence of node $i$. |
| $V_0$ | Walking speed for the young adults group. |
| $V_1$ | Walking speed for the elders and wheelchair users groups. |
| $J_s^{max}$ | Maximum specific flow that does not represent a negative impact on the walking speed. |
| $\rho^k$ | Relationship of occupied area by an evacuee of each group $k \in \mathcal{K}$. |
| $\sigma_{h,k}$ | Parameter defining the initial demand at every origin node $h \in \mathcal{H}$ for the commodity k (0=young,1=Elders, and 2=Wheelchair users). |
| $\psi$ | Penalty for the excess of flow. |
| $\gamma$ | Penalty for not following the age distribution recommendation. |
| $C$ | Maximum number of passengers allowed on a safe boat. |
| $D^k$ | Maximum number of passengers from commodity k allowed in the safe boat. |

The mathematical formulation of the path formulation of the assignment problem in a dynamic network is as follow:

$$\min \zeta + \psi \sum_i^V \sum_t^T f_{i,t} + \gamma \sum_j^\Omega \sum_k^\mathcal{K} \varphi_j^k$$

Subject to:

$$t y_t^l \leq \zeta \text{ for all } l \in \mathcal{L} \text{ and } t \in \mathcal{T}. \tag{5}$$

$$x_{i,n} \leq M y_t^{l1} \text{ for all } P_{i,n}^{l1} \in W_t^{l1} \text{ and } t \in \mathcal{T}. \tag{6}$$

$$z_{i,n}^k \leq M y_t^{l2} \text{ for all } P_{i,n}^{l2} \in W_t^{l2} \text{ , } k \in \mathcal{K} \text{ and } t \in \mathcal{T}. \tag{7}$$

$$\sum_n x_{i,n} = \sigma_{i,0} \text{ for all } i \in \mathcal{H}. \tag{8}$$

$$\sum_n z_{i,n}^k = \sigma_{i,k} \text{ for all } i \in \mathcal{H} \text{ and } k \in \mathcal{K}. \tag{9}$$

$$\sum_n^{\Delta 1_{i,t}} \frac{x_{i,n}}{A_i} V_0 + \sum_n^{\Delta 2_{i,t}} \sum_k^{\mathcal{K}} \frac{z_{i,n}^k}{A_i} V_1 \rho^k \leq J_s^{max} + f_{i,t} \text{ for all } i \in \mathcal{V} \setminus \{Q\} \text{ and } t \in \mathcal{T}. \tag{10}$$

$$\sum_{n}^{\Delta 1_{i,t}} x_{i,n} + \sum_{n}^{\Delta 2_{i,t}} \sum_{k}^{\mathcal{K}} z_{i,n}^{k} = 0 \text{ for all } i \in \mathcal{Q} \text{ and } t \in \mathcal{T}. \quad (11)$$

$$\sum_{(i,n)}^{\omega_j^{l1}} x_{i,n} + \sum_{k} \sum_{(i,n)}^{\omega_j^{l2}} z_{i,n}^{k} \leq C_j \text{ for all } j \in \Omega. \quad (12)$$

$$\sum_{(i,n)}^{\omega_j^{l2}} z_{i,n}^{k} \leq D_j^k + \varphi_j^k \text{ for all } j \in \Omega \text{ and } k \in \mathcal{K}. \quad (13)$$

$$y_t^l \in \{0,1\} \text{ for all } t \in \mathcal{T} \text{ and } l \in \mathcal{L}. \quad (14)$$

$$x_{i,n} \geq 0 \text{ for all } i \in \mathcal{H} \text{ and } n \in P_{i,n}^{l1}. \quad (15)$$

$$z_{i,n}^k \geq 0 \text{ for all } i \in \mathcal{H} \text{ , } n \in P_{i,n}^{l2} \text{ and } k \in \mathcal{K}. \quad (16)$$

$$f_{i,t} \geq 0 \text{ for all } i \in \mathcal{V} \text{ and } t \in \mathcal{T}. \quad (17)$$

$$\varphi_{j,t} \geq 0 \text{ for all } j \in \Omega \text{ and } t \in \mathcal{T}. \quad (18)$$

$$\zeta \geq 0 \quad (19)$$

Constraint (1) makes sure that $\zeta$ takes the value of the latest time step with a positive arrival of flow. Constraints (2) and (3) record the clearing time for paths with positive flow of passengers in the binary variables $y_t^{l1}$ and $y_t^{l1}$, if no path finishing at a given time has a flow the binary variable takes the value of 0, otherwise its value is 1. (4) and (5) are our balance constraints and assure that the demand in each origin node is completely satisfied by the path assignment in each layer. Constraint (6) keeps track of the specific flow of evacuees in the same node $(i,t)$ in both layers, this implies that the congestion in the nodes depends on both layers. Also, it makes sure that we will face a penalty when the maximum specific flow is surpassed. (7) assures that the number of people traversing the forbidden corridors is zero. Constraint (8) makes sure that the total number of people arriving at a given end node $j \in \Omega$ does not exceed the capacity, this capacity is given by the number of lifeboats associated with that node. Constraint (9) keeps track of a desired age distribution within the lifeboats, note that this soft constraint only acts on layer 2 because we do not want to penalize the group of people with the slowest walking speed in case a hard constraint is binding, if the next available lifeboat is to costly to achieve, then it is possible to violate the upper limit imposed to this group. On the other hand, you can fill up freely a lifeboat with young adults. Finally, constraints (10)(11)(12)(13)(14), and (15) define the domain of the decision variables.

# Appendix. B

**Algorithm 1 Network Simplex Inspired Algorithm**

```
Global: PathList                                         ▷ List of computed paths
PathList ← refinePaths(PathListY)                        ▷ Optimize for younger group
PathList ← refinePaths(PathListE)                        ▷ Optimize for elder group
LastArrTime ← GetLastArrival(PathList)
while true do
    Update passenger counts in each space-time node
    Update network topology
    Update paths according to new topology
    SortByTopology
    Sort PathList
    RefinePaths                                          ▷ For younger group
    RefinePaths                                          ▷ For elder group
    NewLastArrTime ← GetLastArrival(PathList)
    if NewLastArrTime ≥ LastArrTime then
        Break                                            ▷ Terminate loop if no further improvement
    else
        LastArrTime ← NewLastArrTime
RefinePaths                                              ▷ For younger group
RefinePaths                                              ▷ For elder group
Return PathList
```

## Appendix. C

**Algorithm 2** OptimizeClearTime()

```
Input: worstPath, tfWorstPath, dagRank, targetLB
GLOBAL: solution                                   ▷ List containing all paths
GLOBAL: congestionThreshold                        ▷ List of congestion interval limits
GLOBAL: fullBoats                                  ▷ List of lifeboats with no available seats

Assess congestion and set a crossroad as the origin node
Compute shortest path using DAG algorithm and define incumbentPath to the same lifeboat
amountOfFlow ← minimum flow required to reach a congestion threshold
Append incumbentPath to solution and sort

if amountOfFlow ≠ tfWorstPath then
    dagRank ← dagRank + 1
    optimizeClearTime                              ▷ Recursive call
else
    Remove worstPath from solution and find a new worstPath
    if worstPath ≠ incumbentPath then
        optimizeClearTime                          ▷ Recursive call
    else
        Check alternative paths on lifeboats with available capacity
        if worstArrivalTime > optional arrival times then
            targetLB ← newLifeboat
            optimizeClearTime                      ▷ Recursive call
        else
            Check alternative paths on lifeboats with full capacity
            if worstArrivalTime ≤ optional arrival times then
                Terminate
```

**Algorithm 2** OptimizeClearTime()

```
while fullBoats ≠ empty do
    Swap origin nodes and targetLB nodes
    Compute shortest path using DAG algorithm and find optional paths
    if worstArrivalTime ≤ optional arrival times then
        Check another lifeboat
    else
        incumbentPath ← optionalPath
        Find minimum flow required to reach a congestion threshold
        Append incumbentPath to solution and sort
        if amountOfFlow ≠ tfWorstPath then
            Check another lifeboat
        else
            Remove worstPath from solution
            Terminate
```

NHH

NORGES HANDELSHØYSKOLE
Norwegian School of Economics

Helleveien 30
NO-5045 Bergen
Norway

**T** +47 55 95 90 00
**E** nhh.postmottak@nhh.no
**W** www.nhh.no