Distributed Evacuation in Graphs with Multiple Exits *

Piotr Borowiecki

Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Poland.

Shantanu Das

Aix Marseille Univ, CNRS, LIF, Marseille, France.

Dariusz Dereniowski

Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Poland.

Łukasz Kuszner

Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Poland.

Abstract

We consider the problem of efficient evacuation using multiple exits. We formulate this problem as a discrete problem on graphs where mobile agents located in distinct nodes of a given graph must quickly reach one of multiple possible exit nodes, while avoiding congestion and bottlenecks. Each node of the graph has the capacity of holding at most one agent at each time step. Thus, the agents must choose their movements strategy based on locations of other agents in the graph, in order to minimize the total time needed for evacuation. We consider two scenarios: (i) the centralized (or offline) setting where the agents have full knowledge of initial positions of other agents, and (ii) the distributed (or online) setting where the agents do not have prior knowledge of the location of other agents but they can communicate locally with nearby agents and they must modify their strategy in an online fashion while they move and obtain more information. In the former case we present an offline polynomial time solution to compute the optimal strategy for evacuation of all agents. In the online case, we present a constant competitive algorithm when agents can communicate at distance two in the graph. We also show that when the agents are heterogeneous and each agent has access to only a subgraph of the original graph then computing the optimal strategy is NP-hard even with full global knowledge. This result holds even if there are only two types of agents.

Keywords: Discrete evacuation, Distributed algorithm, Mobile agents, Network flow

*Research partially supported by the Polish National Science Center grant DEC-2011/02/A/ST6/00201 and by the ANR (France) project MACARON (anr-13-js02-0002).

1 Introduction

Coordinated action of multiple autonomous agents is a subject of study in many contexts. Frequently, the communication capabilities of agents are limited, and the exchange of information between agents is only possible when they are located close to each other, which creates challenges for coordination problems. We consider here the *evacuation* problem which requires multiple mobile entities to reach designated safe area, in a coordinated manner. This can correspond to evacuating a building with a crowd of people inside, who are required to leave the building through emergency exits due to flood, fire, bomb attack, gas leak or other dangers. One could also imagine other scenarios, e.g., a swarm of mobile robots that are required to gather in selected places (exits in the context of evacuation terminology).

In practical situations it can be desired to calculate the evacuation strategy 'on the fly' – just in the time, when the evacuation process is about to start. In fact it is possible to compute a customized solution adapted to the current situation, depending on the number of agents and their locations. Centralized computation of the evacuation strategy has an important drawback as the security of the system relies strongly on the central computing unit and the communication between central unit and agents. Hence, for the safety reasons, a distributed approach could be a better solution. Indeed, recently some attention has been paid to the study of non-centralized evacuation control systems for example assuming a usage of handheld devices (like smartphones) by evacuees [8].

In this paper we try to answer the question of how the lack of knowledge about the positions of other agents may influence the quality of a solution, e.g., by creating bottlenecks when too many agents try to go to the same exit. We compare the distributed (online) solutions to the problem with centralized (offline) solutions. Another important issue for evacuation is when the mobile agents have intrinsic characteristics preventing some of them from visiting some areas of the environment. For example, disabled people might not be able to go through steep stairs inside a building. To address such situations we consider a reachability function defined for each of the agents separately and investigate the coordinated evacuation of such heterogenous agents.

1.1 Related Work

Evacuation models can be classified as macro- and microscopic [14], where macroscopic models are based on optimization approaches of dynamic network flows and do not consider individual characteristics of evacuees while microscopic models are based on simulations in which physical abilities of evacuees are considered.

Indeed, the evacuation problem has been widely described as an application of flows over time (dynamic flows) in time expanded graphs [11, 12] (see the PhD thesis by Jan-Philipp Kappmeier [16] for the recent survey and the plethora of literature references).

Graphs based models are the first choice in the street network modelling [15] but also such environments like buildings [2, 4] or caves [3] can be modelled by graphs. In this case, additional properties of graphs can be considered to reflect properties of the real network such as the transit times, connection capacities, node capacities etc. The motivation for the graph model considered in this paper is lattice based, discretization graph where the Euclidean space is divided into small cells in the shape of squares or hexagons. Such an approach has been mostly considered for the sake of microscopic simulations [16].

Another type of investigations has been done in the context of distributed algorithms for collaborative agents. Chrobak et al. [5] investigate evacuation from the line and Czyżowicz et al. [6, 7] study the problem of evacuation through an unknown exit from a disc shaped continuous space.

The goal in these papers is to minimize the time when the last agent reaches an exit, which is in contrast to a collaborative search problem studied earlier [1], where the goal is to find an exit by the first robot as fast as possible. Many of these results attempt to provide the best algorithms in terms of evacuation time, compared to the obvious optimal solution when the exit is known to the agents. In the above investigations, the number of agents is small and the issue of congestion or collisions does not appear at all.

1.2 Our Results

In this paper, we consider the evacuation problem in the (discrete) graph model where the agents start from distinct nodes and they know the environment (i.e., the graph and the designated exits) but may not know the initial position of other agents. We first consider the centralized or offline version of our problem which we formulate in Section 2 while in Section 3 we present polynomial time algorithms for computing the evacuation strategy in general graphs using the time expansion technique. Section 4 gives a formal statement of the distributed version of the problem when the agents can only communicate locally and thus the strategy must be computed in online fashion. As a first step towards a distributed solution, we consider tree networks and we present and analyze distributed strategies for evacuation from tree networks in Section 5. In particular, in Section 5.1 we prove that there does not exist any distributed algorithm for evacuating agents in less than 2 times the offline optimal t_{opt} steps even in tree networks (Theorem 2). On the other hand, in Section 5.3 we give a distributed algorithm for trees proving its correctness and bounding the evacuation time by $72 \cdot t_{\text{opt}}$ steps (Theorem 4). Finally, in Section 6, we consider the evacuation of heterogenous agents having additional restrictions, namely every agent has access to a predefined subset of edges in the graph (see the similar concept applied to rendezvous problem in [10]). We show that computing the optimal evacuation strategy is NP-hard in this case even if there are only two types of agents and even if the evacuation time is a small constant.
Due to the space constraint, proofs of some of the lemmas and theorems have been omitted and they can be found in the appendix.

2 Problem formulation

In this section we formulate the discrete evacuation problem for the offline setting. Additional assumptions required in the distributed setting will be given in Section 4. In a basic version of the problem we are given a simple graph $G = (V, E)$ with node set V, edge set E, and size $n = |V(G)|$, and the set $\mathcal{A}$ of k agents initially placed on preselected nodes of the graph G, called *homebases* such that no two agents occupy the same homebase. In what follows the set of all homebases is denoted by H. We also distinguish a subset X of nodes called *exits*. Time is divided into *steps* of unit duration. In each step the following actions are performed by each agent:

(A1) an agent either changes its location from the currently occupied node v to one of its neighbors, or remains at v,

(A2) an agent that occupies some node in X, *evacuates* from the graph.

An agent that evacuates is removed from the graph. If in some step an agent is not evacuated, then we say that the agent is *present* in the graph. The node occupied by an agent i at the end of action (A1) in step s is denoted by $\nu_i(s)$, while $\nu_i(0)$ denotes the initial position (the homebase) of the agent. A sequence $(\nu_1, \ldots, \nu_k)$ of the above functions is called an *evacuation strategy* if for

each agent $i \in \mathcal{A}$ there is a step s_i such that $\nu_i(s_i) \in X$, and for each pair of distinct agents $i, j \in \mathcal{A}$ in each step $s \in \{1, \ldots, \min\{s_i, s_j\}\}$ it holds $\nu_i(s) \neq \nu_j(s)$. The *length* of an evacuation strategy is defined as $\max\{s_1, \ldots, s_k\}$. Note that according to (A1) we allow any pair of agents i, j located in neighbouring nodes to move in a single step s in such a way that $\nu_i(s) = \nu_j(s-1)$ provided that $\nu_j(s-1) \neq \nu_j(s)$. The decision version of discrete evacuation problem EVAC is defined as follows:

Problem EVAC

Input: a graph G, an integer l, a set X of exits and a set H of homebases keeping k agents.

Question: does there exist an evacuation strategy of length at most l?

3 The complexity of EVAC

In this section we argue that there exists a polynomial-time algorithm for the problem EVAC. The solution is obtained using classical results for the maximum flow problem. More precisely, for an input (G, l, X, H) of EVAC we construct an *evacuation digraph* $Q = (U, A)$ for which there exists a flow of size k if and only if the answer to EVAC is YES.

Construction 1. Let $V = \{v_1, \ldots, v_n\}$ be the set of nodes of a given graph G and let $Q_0, Q_1, \ldots, Q_l$ be disjoint digraphs such that for each digraph $Q_j = (V_j^{\text{in}} \cup V_j^{\text{out}}, A_j)$ with $V_j^{\text{in}} = \{v_{j,1}^{\text{in}}, \ldots, v_{j,n}^{\text{in}}\}$ and $V_j^{\text{out}} = \{v_{j,1}^{\text{out}}, \ldots, v_{j,n}^{\text{out}}\}$, the arc set A_j is defined as follows $A_j = \{(v_{j,p}^{\text{in}}, v_{j,p}^{\text{out}}) \mid p \in \{1, \ldots, n\}\}$. The node set U of digraph Q consists of the nodes of all digraphs Q_j and the two additional nodes s and t. More formally $U = \{s, t\} \cup \bigcup_{j \in \{0,\ldots,l\}} (V_j^{\text{in}} \cup V_j^{\text{out}})$. To obtain the arc set A of Q we take all arcs of digraphs $Q_0, Q_1, \ldots, Q_l$, and for each $j \in \{0, \ldots, l-1\}$ between the nodes of Q_j and Q_{j+1} we add the arcs: $(v_{j,p}^{\text{out}}, v_{j+1,p}^{\text{in}})$ for each $p \in \{1, \ldots, n\}$, and $(v_{j,p}^{\text{out}}, v_{j+1,q}^{\text{in}})$ if for the corresponding nodes v_p, v_q of the graph G it holds $\{v_p, v_q\} \in E(G)$. We also add to A the following arcs: $(s, v_{0,p}^{\text{in}})$ if the corresponding vertex v_p of G belongs to H, and $(v_{j,p}^{\text{out}}, t)$ whenever v_p belongs to X and $j \in \{0, \ldots, l\}$. Less formally, we add arcs outgoing from s to the nodes in V_0^{in} corresponding to all homebases of agents, and we add arcs incoming to t from all nodes in $V_0^{\text{out}} \cup \cdots \cup V_l^{\text{out}}$ corresponding to the exits. All arcs in Q are assumed to have unit capacities. Clearly, for every (G, l, X, H) the corresponding evacuation digraph can be constructed in polynomial time. □

Lemma 1. *There exists an s-t flow of size k in evacuation digraph Q if and only if the answer to* EVAC *is* YES *for the input (G, l, X, H).*

Proof. Suppose that we have an s-t flow of size k in the digraph Q. Clearly, such a flow is made of k s-t paths that have only their endpoints in common. Each such a path P encodes the moves of a corresponding agent in an evacuation strategy. Indeed, for each $j \in \{0, \ldots, l-1\}$, the path P contains exactly one node $v_{j,p}^{\text{out}}$ in V_j^{out} and exactly one node $v_{j+1,q}^{\text{in}}$ in V_{j+1}^{in}, and $(u, v) \in A$. Then, the evacuation strategy dictated by P moves the corresponding agent from v_p to v_q in G in step $j+1$. Since the internal nodes of the paths that constitute the flow have no internal nodes in common, the evacuation strategy obtained in this way is guaranteed to have no 'collisions' between agents. Also, since the only arcs incoming to t are outgoing from nodes in V_j^{out} that correspond to the exit set X, it is ensured that each agent reaches an exit within the time limit l. □

Since the maximum flow problem is polynomial [9] we have the following.

Theorem 1. *There exists a polynomial-time algorithm for solving the problem* EVAC. □

4 Distributed evacuation model

We now consider the distributed version of the evacuation problem where each agent must autonomously compute its strategy to move based on local information and communication with the agents it encounters. As before the model is synchronous and during each time step, each agent first communicates with other agents and performs local computations to decide on the action to be performed in this step; then the agent performs the action (i.e. it moves, stays or exits). Recall that the two possible actions are (A1) and (A2) defined in Section 2. Since each node can hold at most one agent at a given time, the actions selected by the agents in each time step must satisfy this restriction. However it is possible for two agents in adjacent nodes to swap their positions in a given step (there are no capacity restrictions on edges of the graph in our model).

Our next assumption is that each agent knows in advance the network and its own homebase. This includes the knowledge of the location of all exits. The nodes of the network have unique identifiers (thus the agents also have unique identifiers as each agent may 'inherit' the identifier of its homebase). Each agent has the information necessary for navigating in the graph i.e. the agent knows which edge to follow to reach the node selected for its next location. However the locations and the number of other agents are initially unknown to the agents.

The communication model is as follows. Each agent can directly communicate with any other agent that is within a distance of at most 2 in the graph. Note that the exchange of messages within distance of two (not just one) is a necessary assumption that follows from the necessity of avoiding collisions between agents. Otherwise two agents at distance two, being unaware of each other may decide to move to the same node. We assume that communication is instantaneous (or it takes negligible time compared to the duration of a time step). Thus, at the beginning of each step, any two agents can exchange any number of messages if the distance between them is not larger than 2. Agents can use message passing to communicate indirectly with other agents via intermediate agents. For example, see Figure 1 where the agents form two 'groups' of communicating agents such that within each group any two agents i_1 and i_j can communicate either directly (if they are within the distance of two) or indirectly if i_1 and i_j are at distance greater than two but there exist agents $i_2, \ldots, i_{j-1}$ such that for each $p \in \{1, \ldots, j-1\}$ the agents i_p and i_{p+1} are at distance at most 2. In this example, agents i_1 and i_4 can communicate by passing messages through agents i_2 and i_3. Since we do not impose any restrictions on the amount of messages exchanged between any

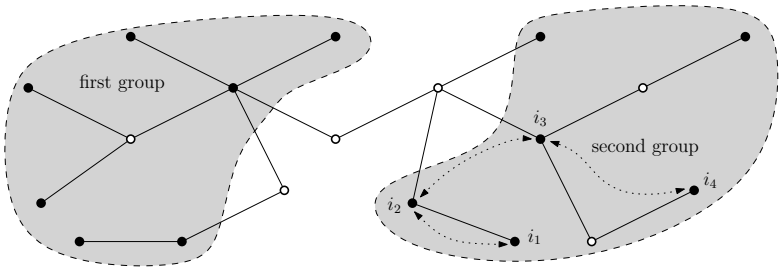

Figure 1: Example graph with two groups of communicating agents. Black nodes contains agents and white nodes are unoccupied.

two agents at the beginning of each step, we may assume without loss of generality that each agent begins each step by identifying positions of all agents with whom it may communicate directly or indirectly. This can be achieved by performing a broadcast algorithm by the agents and since this

part is straightforward, we omit the details and assume also in our algorithms that each step begins with collecting information about all agents that can be reached directly or indirectly by an agent in that step.

5 Evacuation in tree networks

In this section we investigate evacuation from tree networks in the distributed setting as formulated above. The performance measure we use for a distributed algorithm is the *competitive ratio*, defined as the worst case ratio of the evacuation time achieved by the algorithm over t_{opt} the optimal evacuation time in the offline setting for the same instance (We consider the worst case scenario over all tree networks and all possible locations of homebases). We start by showing a lower bound on the competitive ratio of any distributed strategy, in Section 5.1. Then, in Section 5.2, we introduce some notations used in Section 5.3 which provides a distributed evacuation algorithm for trees.

5.1 A lower bound

Theorem 2. *The competitive ratio of any distributed evacuation algorithm is at least* 2 *even for trees.*

Proof. Given two integers $k, p \geq 1$, let us consider a tree with $3kp + 1$ nodes: a node v_0 and $3kp$ nodes $v_{i,j}$, where $i \in \{1, 2, \ldots, k\}$ and $j \in \{1, 2, \ldots, 3p\}$. There are $k + 1$ exits, $X = \{v_0, v_{1,3p}, v_{2,3p}, \ldots, v_{k,3p}\}$. Node v_0 is adjacent to k nodes: $v_{1,1}, v_{2,1}, \ldots, v_{k,1}$ and for every $i \in \{1, 2, \ldots, k\}$ and $j \in \{1, \ldots, 3p - 1\}$ there is an edge $\{v_{i,j}, v_{i,j+1}\}$. There are k agents $a_1, \ldots, a_k$, where $\nu_{a_i}(0) = v_{i,p}$.

Informally speaking, there are k paths of length $3p$ joined by an exit node v_0, with k agents, one in each path, placed p steps away from the exit v_0 and $2p$ steps from the other exit $v_{i,3p}$ at the end of the path. We take $k = 4p$. Note that the optimum evacuation time for this instance is $t_{\text{opt}} = 2p$.

Let A be any distributed algorithm for Evac. Let us consider the step $2p - 2$ of the execution of A. There are two cases:
(i) There exists $i \in \{1, 2, \ldots, k\}$ such that there is an agent a on the path connecting $v_{i,p+1}$ and $v_{i,3p-2}$. Clearly $a = a_i$. Note that the agent a_i was unable to reach the node $v_{i,1}$ adjacent to the exit v_0 and thus this agent performed no communication with another agent within the first $2p - 2$ steps. If so, let us consider a different instance of the problem in which the graph is the same as in the former one but there exists only one agent, namely a_i with the same homebase $v_{i,p}$. Thus, the input to the algorithm executed by the agent a_i is the same as in the previous scenario. Therefore, due to the lack of communication as argued above, the behaviour of the agent a_i is exactly the same in the latter scenario as in the former one. Thus, the agent a_i will not evacuate before step $2p$. The evacuation time in the latter scenario with single agent present is clearly p and hence in this case the competitive ratio is 2.

(ii) In the second case, every agent a_i is somewhere on the path $v_{i,1}, v_{i,p}$ or already evacuated during the first $2p - 2$ steps. If a_i evacuated, then clearly the exit it used is v_0. We consider two additional subcases. In the first subcase, all agents evacuate through v_0. Since $k = 4p$ and $t_{\text{opt}} = 2p$, the competitive ratio is 2 as required. In the second subcase, some agent a_i evacuates through the exit $v_{i,3p}$. But then, a_i needs to traverse the path connecting $\nu_{a_i}(2p - 2)$ and $v_{i,3p}$ in steps that follow the step $2p-2$. This path contains the path of length $2p$ connecting $v_{i,p}$ with $v_{i,3p}$. Thus, a_i does not evacuate within the first $(2p - 2) + 2p$ steps. Therefore, in the second subcase

the competitive ratio we obtain is $\frac{4p-2}{2p}$ which can be made arbitrarily close to 2 by taking p large enough. □

5.2 Additional notations

The distance between a pair of nodes u and v denoted by $d(u, v)$ is the length of the path (i.e., the number of edges) connecting these nodes. By the distance between two agents i and j in step s we mean the distance between the nodes in which the agents are located at the end of step s, i.e., $d(\nu_s(i), \nu_s(j))$.

The exit *associated with node* v is the exit $x \in X$ with the smallest identifier among the exits at minimum distance from v. By the *primary exit* of agent i, denoted by $\mathrm{pe}(i)$, we mean the exit associated with $\nu_i(0)$, the homebase of the agent. Clearly, there may be many agents having the same node as the primary exit but for each agent the primary exit is unique. Also note that since the primary exit depends only on the homebase of the agent, it remains the same even if agent changes its position during evacuation. As we will see later, in the early stage of our algorithm each agent attempts to evacuate through its primary exit. This leads to the formation of groups of agents having the same primary exits and allows for computation of group rather than individual strategies. We address this issue in more detail during analysis of our algorithm.

For the description of the algorithm we also need a specific partition $\mathcal{V} = (V_{x_1}, \ldots, V_{x_\eta})$ of the node set of a given tree T with the set of exits $X = \{x_1, \ldots, x_\eta\}$. Namely, for each exit $x \in X$ the set V_x of $\mathcal{V}$ is defined as a set consisting of all nodes having x as the associated exit. Naturally, $\bigcup_{x \in X} V_x = V(T)$ and since primary exits are uniquely determined, for any pair of distinct exits x_p, x_q it holds $V_{x_p} \cap V_{x_q} = \emptyset$. It is also not hard to see that for each $x \in X$ the subgraph induced by V_x is connected. Let T_x denote the tree induced by V_x.

At the beginning of its computation each agent roots the tree T at the node with the smallest identifier. This ensures that all agents select the same root. Without loss of generality assume that agents rooted T in node v_r that belongs to the tree T_{x_r} which corresponds to exit x_r (see in Fig.2(a) where $r = 2$). According to the above assumption we construct a tree $\widetilde{T}$ with the node set X, root x_r, and edge between each pair of nodes x_p, x_q for which T contains an edge $\{v, u\}$ such that $v \in V_{x_p}$ and $u \in V_{x_q}$, $p, q \in \{1, \ldots, \eta\}$ (see Fig.2(b) with x_1, x_4 corresponding to v, u in Fig.2(a)). Following "away from the root" natural orientation of the edges of $\widetilde{T}$ one can easily relate subtrees T_x of T. Namely, for any two exits x_p, x_q we say that T_{x_p} is a *child* of T_{x_q} if x_q is the parent of x_p under the above orientation of $\widetilde{T}$.

The moves of agents in the early stage of our algorithm result in grouping the agents at exits. To address this more accurately let $x \in X$ and let s be a step. A *group of agents at* x *in step* s, denoted by $\mathcal{G}_{x,s}$ is a maximal set of agents for which $\nu_i(s) \in V(T_x)$ and each node of the path connecting $\nu_i(s)$ and x is occupied by an agent form $\mathcal{G}_{x,s}$. We say that an agent i *joins the group* $\mathcal{G}_{x,s}$ *in step* s if $i \notin \mathcal{G}_{x,s-1}$ but $i \in \mathcal{G}_{x,s}$. An agent i *joins exit* x in step s if in step s it joins the group $\mathcal{G}_{x,s}$ (we allow that the group is empty before step s).

Finally, let $\mathcal{A}_x$ denote the set of agents with their homebases in V_x, i.e., $\mathcal{A}_x = \{i \in \mathcal{A} \mid \mathrm{pe}(i) = x\}$. Note that $\mathcal{A}_x$ is not necessarily a group at x.

5.3 The algorithm

Our algorithmic result obtained in this section is achieved in two steps. First, we give an evacuation algorithm that receives as an input an upper bound B on the optimum time required for the evacuation t_{opt} and we prove that this algorithm has a constant competitive ratio. Then, we

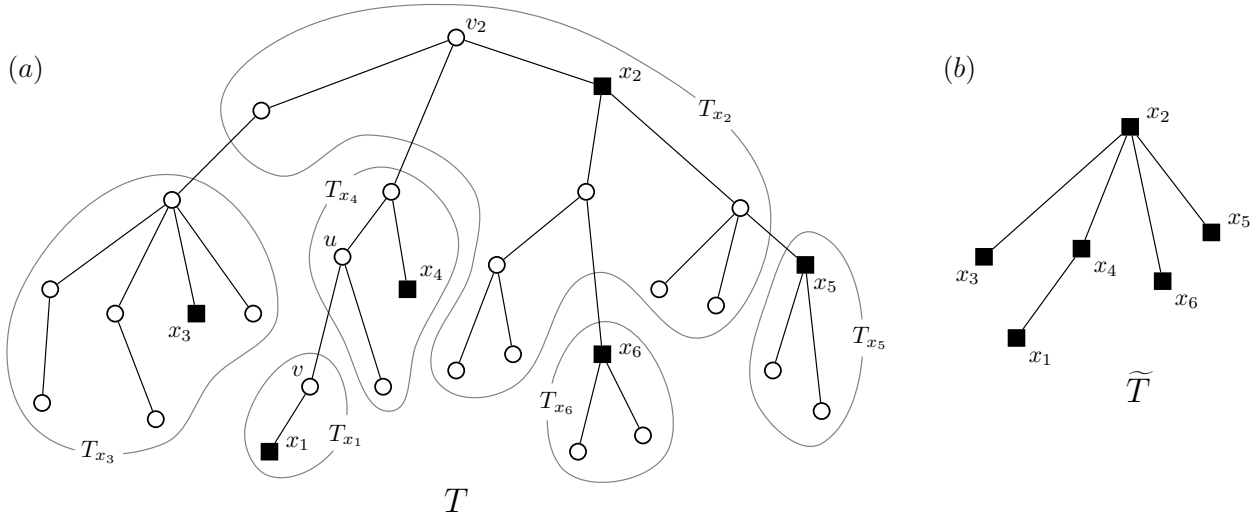

Figure 2: Tree T with exits $x_1, \ldots, x_6$, subtrees $T_{x_1}, \ldots, T_{x_6}$, and the corresponding tree $\widetilde{T}$.

provide our main procedure that uses the 'doubling technique' (the first algorithm mentioned above is called several times with exponentially increasing values on possible upper bounds) to disregard the assumption that an upper bound on evacuation time is known. As we prove, the competitive ratio of the second algorithm increases by a constant with respect to the first one. Both algorithms are formulated for an executing agent i.

Our first algorithm called `BoundedDTE` (*Bounded Distributed Tree Evacuation*). We start with an informal description providing the key ideas. The algorithm can be seen as having two phases. In the first phase, the agents located in the subtree composed of nodes in V_x want to communicate with each other to establish the best strategy for them. This strategy that they aim to establish ignores possible agents located outside of V_x. However, the homebases in V_x are possibly spread in such a way that the communication between the agents is not possible initially. Thus, all agents have to meet and they do so by going to x and forming a group at x. It can be estimated when all agents joined the group at x thanks to the fact that an upper bound B on t_{opt} is given as an input. Once all agents joined the group (we point out that at this point some agents possibly evacuated through x) they compute a strategy in which some of the agents remain to evacuate through x while some agents will follow to exit x' such that $T_{x'}$ is a child of T_x. Then the second phase starts in which each agent goes to the assigned exit.

In the pseudocode we will use the phrase *if possible* to indicate that the executing agent i verifies if the specified move can be performed. In particular, if the agent i wants to move to a node u, then communication with all agents located at u and its neighbors is required. This communication is needed to determine all agents that want to move to u in this given round and if there is more than one such an agent, then the agents decide which one (as there can be no more than one) will perform the move. In our algorithm such ties can be resolved arbitrarily.

Theorem 3. *If B is an upper bound on the evacuation time t_{opt}, then Procedure* `BoundedDTE` *evacuates agent i in $9 \cdot B$ steps.*

We now proceed to the description of our main algorithm `DTE` (*Distributed Tree Evacuation*) in which we disregard the previous assumption that an upper bound on t_{opt} is known to agents,

Procedure 1 `BoundedDTE`(T, B) for agent i

Input: the underlying tree T, the time bound B.

1: **for** $k = 1$ **to** B **do**
2: If possible, then move to the adjacent node that is closer to pe(i).
3: **end for**
4: Compute the optimum evacuation strategy for the group $\mathcal{G}_{\mathrm{pe}(i),B}$ taking into account pe(i) and exits in child trees of $T_{\mathrm{pe}(i)}$. Let *secondary exit*, se(i), be the exit for agent i computed in this strategy.
5: **for** $k = 1$ **to** $8 \cdot B$ **do**
6: If possible, then move to the adjacent node that is closer to se(i).
7: **end for**

and again we start with an informal description. The algorithm, as indicated earlier, proceeds by guessing the possible upper bounds B on t_{opt}. For each choice of B, `DTE` makes a call to `BoundedDTE` with input B. If the value of B is indeed at least t_{opt}, then all agents will successfully evacuate. Otherwise, the agents who still did not evacuate reverse their movements to return to their homebases.

Procedure 2 `DTE`(T) for agent i

Input: the underlying tree T.

1: $B \leftarrow 1$
2: **while** true **do**
3: $B \leftarrow 2 \cdot B$
4: **if** $d(\nu_i(0), \mathrm{pe}(i)) > B$ **then**
5: stay immobilized for next $18 \cdot B$ steps.
6: **else**
7: Apply Procedure `BoundedDTE` for $9 \cdot B$ steps or until the evacuation of i.
8: reverse $9 \cdot B$ steps made according to Procedure `BoundedDTE` reaching its homebase.
9: **end if**
10: **end while**

Theorem 4. *Procedure* `DTE` *evacuates all agents in* $\Theta(t_{\mathrm{opt}})$ *steps.*

Proof. Once the variable B achieves value B' which is larger than t_{opt} ($t_{\mathrm{opt}} < B' \leq 2 \cdot t_{\mathrm{opt}}$) then Procedure `DTE` eventually evacuates all remaining agents. Let us estimate t_2, the number of steps required for the execution of Procedure `DTE`:

$$t_2 \leq 18 \cdot (2 + 4 + \cdots + B') \leq 36 \cdot B' \leq 72 \cdot t_{\mathrm{opt}}.$$

Now it is required to justify that agents are able to realize the described strategy. The analysis is very similar to that of Procedure `DTE` but there are two possibilities of additional traffic jams that must be considered.

First, it can happen that agents from the group $\mathcal{G}_{\mathrm{pe}(i),B}$ executing statements described in lines: 7, 8 of Procedure `DTE` encounter immobilized agents from the rest of $\mathcal{A}_{\mathrm{pe}(i)}$. In this case a swap of agents will be applied. Suppose that traveling agent a at node u is about to swap with immobilized agent j at node v at the path uvw. Then, a swaps with j and they move: a moves from v to w and j

moves from u back to v. Observe here also, that during the reverse phase (line 8) of Procedure `DTE` immobilized agents are still immobilized in the same place, thus making the reverse phase possible.

An interaction in between agents from $\mathcal{G}_{\mathrm{pe}(i),B}$ and agents from the parent subtree of $T_{\mathrm{pe}(i)}$ is covered by the correctness proof of Procedure `DTE`. □

6 Generalizations of Evac

In this section we introduce the problem RestEvac, a generalization of Evac in which the moves of agents are restricted so that each agent has access to a preselected subset of edges.

6.1 Evacuation with restricted access to edges

We start by defining an evacuation strategy in this scenario. All restrictions of Evac carry over, except that additionally for each agent $i \in \mathcal{A}$, a set $R_i \subseteq E(G)$ of *permitted edges* is given as a part of the input, while action (A1) defined in Section 2 is replaced by:

(RA1) each agent either changes its location from the currently occupied node v to one of its neighbors u provided that $\{u, v\} \in R_i$, or the agent remains at v.

A sequence of functions $(\nu_1, \ldots, \nu_k)$ that satisfy (RA1) and (A2) is called *restricted evacuation strategy*. The decision version of our new problem, called RestEvac can be defined as follows:

Problem RestEvac

Input: a graph G, an integer l, a set X of exits, a set H of homebases keeping k agents, and sets $R_1, \ldots, R_k$ specifying permitted edges.

Question: does there exist a restricted evacuation strategy of length at most l?

6.2 The complexity of RestEvac – restricted length

We now show that the RestEvac problem is NP-hard by reducing the NP-complete 3-dimensional matching problem, denoted by 3DM, to RestEvac. We first recall the former problem. The input to 3DM consists of three pairwise disjoint m-element sets A, B, C and a set of triples $M \subseteq A \times B \times C$. The answer to 3DM is yes if and only if there exists $M' \subseteq M$ such that $|M'| = m$ and for every two distinct triples (a, b, c) and (a', b', c') in M' it holds $a \neq a'$, $b \neq b'$ and $c \neq c'$.

Theorem 5. *The problem* RestEvac *is* NP*-complete even if the input parameter l equals* 2*.*

Proof. We describe our reduction by constructing the input to RestEvac. Suppose that (A, B, C, m, M) is the input to 3DM. For any triple $(a, b, c) \in M$ define for brevity $\xi((a, b, c)) = \{\{a, b\}, \{b, c\}\}$. Let G be defined as follows:

$$G = \left(A \cup B \cup C, \bigcup_{Z \in M} \xi(Z) \right).$$

Set $l = 2$, $X = C$, $k = m$ and let the k agents be initially placed on the nodes in A, i.e., set $H = A$. For each agent $i \in \mathcal{A}$ having its homebase $a_i \in A$, let

$$R_i = \bigcup_{Z \in N} \xi(Z), \text{ where } N = M \cap (\{a_i\} \times B \times C).$$

This completes the construction and it remains to prove its correctness. We argue that the answer to 3DM is YES if and only if the answer to RESTEVAC is YES.

First, suppose that the answer to 3DM is YES and let $M' \subseteq M$ be the corresponding solution. We construct a restricted evacuation strategy $(\nu_1, \ldots, \nu_k)$ as follows. For each $i \in \mathcal{A}$ there exists $(a_i, b_i, c_i) \in M'$ and we define:

$$\nu_i(0) = a_i, \quad \nu_i(1) = b_i, \quad \nu_i(2) = c_i.$$

It follows from the construction of the input to 3DM and from the definition of X that $\nu_i(0) \neq \nu_j(0)$ for $i \neq j$. Since M' is a solution to 3DM, we obtain that $\nu_i(s) \neq \nu_j(s)$ for $i \neq j$ and $s \in \{1, 2\}$. Consider an edge $e = \{\nu_i(s), \nu_i(s+1)\}$ traversed by agent i in step $s \in \{0, 1\}$. We have that $e \in \xi((a_i, b_i, c_i))$ and hence $e \in R_i$. Finally, $\nu_i(2) \in X$ by construction for each $i \in \mathcal{A}$, which proves that $(\nu_1, \ldots, \nu_k)$ is indeed a restricted evacuation strategy of length 2.

Next, suppose that the answer to RESTEVAC is YES and let $(\nu_1, \ldots, \nu_k)$ be a restricted evacuation strategy of length 2. Since the homebases of agents are in A and $X = C$, we have that for each agent $i \in \mathcal{A}$ it holds $\nu_i(0) \in A$, $\nu_i(1) \in B$ and $\nu_i(2) \in C$. Define $M' = \{(\nu_i(0), \nu_i(1), \nu_i(2)) \mid i \in \mathcal{A}\}$. We have that $\{\nu_i(0), \nu_i(1)\} \in R_i$ and $\{\nu_i(1), \nu_i(2)\} \in R_i$, which for each agent $i \in \mathcal{A}$ implies $(\nu_i(0), \nu_i(1), \nu_i(2)) \in M$. Thus M' is a solution to 3DM. □

6.3 The complexity of RESTEVAC – two types of agents

We now prove that RESTEVAC is computationally hard even if there are only two types of agents, i.e., when there exist two subsets $E_1, E_2 \subseteq E(G)$ of permitted edges such that for each agent $i \in \mathcal{A}$ it holds $R_i \in \{E_1, E_2\}$. Our proof is by a reduction from the problem $(3, 4)$-SAT, in which a logical formula $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ over variables $\mathbf{x}_1, \ldots, \mathbf{x}_p$ is a part of the input, where each clause C_i is a disjunction of exactly three literals and each variable occurs at most four times in the logical formula. The question in $(3, 4)$-SAT is whether there exists a Boolean assignment to variables that satisfies F. The problem $(3, 4)$-SAT is known to be NP-complete [17]. The main result of this section is that RESTEVAC is computationally hard even for fixed length evacuation strategies and only two types of agents whose respective sets of permitted edges are disjoint.

Theorem 6. *The problem* RESTEVAC *is* NP*-complete even if the input parameter l equals* 5 *and for each agent $i \in \mathcal{A}$ it holds $R_i \in \{E_1, E_2\}$, where $E_1 \cap E_2 = \emptyset$.*

7 Conclusions and Open problems

The goal of this work was to introduce a natural distributed model for the discrete evacuation problem and to analyze its basic properties by looking at its complexity and solvability in the mobile agent setting. One assumption that we made in this model, which greatly affects the algorithmic approach, is that local computations of agents and passing of messages take negligible time with respect to the movements of agents. However, one can consider scenarios in which the amount of communication performed in each synchronous step is somewhat more restricted. For example, one could analyze the number of messages exchanged by agents besides the evacuation time.

Another research direction could lead towards dropping the assumption that the network is known to agents. In such scenario, an algorithmic approach should adopt some concepts from the well studied exploration problems for unknown networks. As an intermediate scenario, one could

consider providing the agents only partial information about the network by performing, e.g., a quantitative analysis of the amount of input information (we refer here to the advice complexity introduced in [13]).

In terms of the competitive ratio achievable by any online algorithm, we studied tree networks, and it is interesting to see how this parameter would behave in other network topologies. For example, are there networks in which the competitive ratio is not constant, e.g., a function of the number of agents or some other input parameter? Another open question worth investigating is what happens when the agents cannot communicate but have only local visibility.

References

[1] B. Alspach. Searching and sweeping graphs: a brief survey. *Le Matematiche (Catania)*, 59:5–37, 2004.

[2] G.N. Berlin. The use of directed routes for assessing escape potential. *Fire Technology*, 14(2):126–135, 1978.

[3] R.L. Breisch. An intuitive approach to speleotopology. *Southwestern Cavers*, 6:72–78, 1967.

[4] L.G. Chalmet, R.L. Francis, and P.B. Saunders. Network models for building evacuation. *Fire Technology*, 18(1):90–113, 1982.

[5] M. Chrobak, L. Gasieniec, T. Gorry, and R. Martin. Group search on the line. In *SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science*, pages 164–176, 2015.

[6] J. Czyżowicz, L. Gasieniec, T. Gorry, E. Kranakis, R. Martin, and D. Pajak. Evacuating robots via unknown exit in a disk. In *DISC 2014: Distributed Computing - 28th International Symposium*, pages 122–136, 2014.

[7] J. Czyżowicz, K. Georgiou, E. Kranakis, L. Narayanan, J. Opatrny, and B. Vogtenhuber. Evacuating robots from a disk using face-to-face communication (extended abstract). In *CIAC 2015: Algorithms and Complexity: 9th International Conference*, pages 140–152. Springer International Publishing, 2015.

[8] A. Avilés del Moral, M. Takimoto, and Y. Kambayashi. Distributed evacuation route planning using mobile agents. *Trans. Computational Collective Intelligence*, 17:128–144, 2014.

[9] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972.

[10] A. Farrugia, L. Gasieniec, Ł. Kuszner, and E. Pacheco. Deterministic rendezvous in restricted graphs. In *SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science*, pages 189–200, 2015.

[11] L.R. Ford and D.R. Fulkerson. Constructing maximal dynamic flows from static flows. *Oper. Res.*, 6(3):419–433, June 1958.

[12] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 1962.

[13] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Oracle size: a new measure of difficulty for communication tasks. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 179–187, 2006.

[14] H.W. Hamacher and S.A. Tjandra. Mathematical Modelling of Evacuation Problems – A State of the Art. In *Pedestrian and Evacuation Dynamics*, pages 227–266, Berlin, 2002. Springer.

[15] Y. Higashikawa, M.J. Golin, and N. Katoh. Multiple sink location problems in dynamic path networks. *Theor. Comput. Sci.*, 607:2–15, 2015.

[16] J.-P. Kappmeier. *Generalizations of flows over time with applications in evacuation optimization.* PhD thesis, Technische Universität Berlin, 2015.

[17] C.A. Tovey. A simplified NP-complete satisfiability problem. *Discr. Appl. Math.*, 8:85–89, 1984.

APPENDIX

A Proof of Theorem 3

We prove two lemmas that imply Theorem 3.

Lemma 2. *If B is an upper bound on the evacuation time t_{opt}, then the first loop in Procedure* `BoundedDTE` *(lines: 1–3) finishes in B steps resulting in every agent either evacuated or joined the group at its primary exit.*

Proof. Let D be the largest distance from agent's homebase to its primary exit,

$$D = \max_{i \in \mathcal{A}} d(\nu_i(0), \text{pe}(i)) = \max_{i \in \mathcal{A}} \min_{x \in X} d(\nu_i(0), x).$$

Clearly, we have $D \leq t_{\text{opt}}$.

Now we show that if agent i joins exit x in step s, then $s \leq d(\nu_i(0), \text{pe}(i))$. Let $P(s') \subseteq T$, for $s' \in \{0, 1, \ldots, s\}$ be a maximum subpath of a path between $\nu_i(s')$ and $\text{pe}(i)$ such that $\nu_i(s') \in V(P)$ and for every node $v \in V(P(s'))$ there exists an agent j such that $\nu_j(s') = v$. Agent i joins exit $\text{pe}(i)$ in step s when either $d(\nu_i(s), \text{pe}(i)) = 0$ or $d(\nu_i(s), \text{pe}(i))$ is equal to the length of $P(s)$.

Now, let us observe that in each step s' before i joins $\text{pe}(i)$, the agent i either gets one step closer to $\text{pe}(i)$ and the length of $P(s'+1)$ is the same as the length of $P(s')$ or i does not move, i.e., $\nu_i(s') = \nu_i(s'+1)$ and the length of path $P(s'+1)$ is larger than the length of $P(s')$ by at least 1. □

Lemma 3. *Suppose that B is an upper bound on t_{opt} and that an executing agent did not evacuate within the first B steps. Then, the agent evacuates during the first $4(B + t_{opt})$ iterations of the second loop in Procedure* `BoundedDTE` *(lines: 5–7).*

Proof. Let $\text{evac}(\mathcal{G}_{x,B})$ be the time needed for the evacuation of the group $\mathcal{G}_{x,B}$ according to the strategy computed in step B (line 4 of Procedure `BoundedDTE`).

We first observe that taking into account exits from an indirect descendant (descendant but not child) subtrees of $T_{\text{pe}(i)}$ will not allow faster evacuation. To the contrary suppose that there exists an evacuation strategy such that the evacuation with the usage of such exits shortens the computed evacuation time. Among such strategies let us select one with the smallest number of agents (which must be nonzero) assigned to an exit in an indirect descendant subtree T_{x_q}. Among such agents, let us consider agent a assigned to evacuate through x_q. Being an indirect descendant subtree means that there exists an exit x_p such that T_{x_q} is descendant of T_{x_p} and T_{x_p} is a child of $T_{\text{pe}(i)}$. Now let us consider two paths: P_1 from $\nu_a(B)$ to x_p and P_2 from $\nu_a(B)$ to x_q. We have $P_1 \cap T_{\text{pe}(i)} = P_2 \cap T_{\text{pe}(i)}$. Due to the existence of the bottleneck in between $T_{\text{pe}(i)}$ and T_{x_p} we can redirect the considered agent a to exit x_p without increase of evacuation time, a contradiction.

Let $\text{evac}^T(\mathcal{G}_{x,s})$ be the optimum evacuation time of the group $\mathcal{G}_{x,s}$ through all possible exits in T and let us consider $\text{evac}^T(\mathcal{G}_{\text{pe}(i),B})$. There exists a bottleneck (exactly one edge) in between $T_{\text{pe}(i)}$ and the parent subtree of $T_{\text{pe}(i)}$, so using all exits in T makes it possible to evacuate only one additional agent per step during the evacuation process. On the other hand $\text{evac}(\mathcal{G}_{\text{pe}(i),B}) \leq |\mathcal{G}_{\text{pe}(i),B}|$ as exactly one agent per step can evacuate through $\text{pe}(i)$ and possibly some more agents from $\mathcal{G}_{\text{pe}(i),B}$ through other exits in a child of $T_{\text{pe}(i)}$. Thus

$$\text{evac}(\mathcal{G}_{\text{pe}(i),B}) \leq 2 \cdot \text{evac}^T(\mathcal{G}_{\text{pe}(i),B}). \tag{1}$$

Let $\mathrm{evac}_0(\mathcal{A}_x)$ be the evacuation time for the group $\mathcal{A}_x$ of agents located in their homebases. Let us now bound $\mathrm{evac}_0(\mathcal{A}_{\mathrm{pe}(i)})$. Clearly $\mathcal{G}_{\mathrm{pe}(i),B} \subseteq \mathcal{A}_{\mathrm{pe}(i)}$. As agents from $\mathcal{G}_{\mathrm{pe}(i),B}$ are able to come back to their homebases in B time units reversing the steps taken in the first part of Procedure `BoundedDTE` (lines: 1–3) and the evacuation time for the subgroup must not exceed the evacuation time for the whole group, we have

$$\mathrm{evac}^T(\mathcal{G}_{\mathrm{pe}(i),B}) \leq B + \mathrm{evac}_0(\mathcal{A}_{\mathrm{pe}(i)}) \leq B + t_{\mathrm{opt}}. \tag{2}$$

If the paths of two agents cross in such a way that they find themselves on two adjacent nodes and they want to exchange their positions. They exchange the memory states, i.e., the two agents *swap* without performing actual moves.

Till now, considering the evacuation strategy for the group $\mathcal{G}_{x,B}$ we have ignored the existence of agents from other groups and possible interactions between them. Thus, $\mathrm{evac}(\mathcal{G}_{x,B})$ – the time computed for the evacuation of $\mathcal{G}_{x,B}$ can be different than the time required for the execution of Procedure `BoundedDTE` (lines: 5–7) for the same group. Let us denote the latter by $\mathrm{exec}(\mathcal{G}_{x,B})$.

We will argue that the existence of such groups will not increase the evacuation time according to the strategy computed by Procedure `BoundedDTE` by more than by the factor of two. In fact we will show that

$$\mathrm{exec}(\mathcal{G}_{x,B}) \leq \mathrm{evac}(\mathcal{G}_{x,B}) + \max\{y \in X \mid \mathrm{evac}(\mathcal{G}_{y,B})\}. \tag{3}$$

First, let us observe that the evacuation strategy computed for those agents from a group $\mathcal{G}_{x_r,B}$ which will evacuate through the exit x_r in T_{x_r}, that is $\mathcal{A}^{x_r} = \{a \in \mathcal{G}_{x_r,B} \mid \mathrm{pe}(a) = \mathrm{se}(a) = x_r\}$, will not be affected by any other group and computed evacuation time for all of those agents will be preserved in the execution of Algorithm 1:

$$\mathrm{exec}(\mathcal{A}^{x_r}) = \mathrm{evac}(\mathcal{A}^{x_r}).$$

Now, let us consider any other group evacuating through any particular exit $x \neq x_r$ as its primary and secondary exit: $\mathcal{A}^x = \{a \in \mathcal{G}_{x,B} \mid \mathrm{pe}(a) = \mathrm{se}(a) = x\}$. Those agents' strategies can be affected only by agents from $\mathcal{A}^{yx} = \{a \in \mathcal{G}_{y,B} \mid \mathrm{pe}(a) = y \,\mathrm{and}\, \mathrm{se}(a) = x\}$, where y is the parent exit of x. Clearly, $|\mathcal{A}^{yx}| \leq \mathrm{evac}(\mathcal{G}_{y,B})$, thus Equation (3) holds.

Combining inequalities (1)–(3) and taking the maximum over all possible exits we get the requested bound. □

B Proof of Theorem 6

In order to prove the above theorem we give a construction of the graph G, being a part of the input to RestEvac, then we formulate some helpful observations used in the proofs of Lemmas 4 and 5.

Construction 2. First, on the basis of formula F and the integer p we construct an input graph G for the problem RestEvac. To that end we first define a *variable component* G'. Let the set of the nodes of the component G' be given by

$$\begin{aligned} V(G') &= \left\{v_i^j \mid i \in \{0,\ldots,5\}, j \in \{1,2\}\right\} \cup \left\{u_i^j \mid i \in \{1,\ldots,5\}, j \in \{1,2\}\right\} \\ &\cup \left\{x_{i,i'}^j, y_{i,i'}^j \mid i \in \{1,\ldots,4\}, i' \in \{0,\ldots,4\}, j \in \{1,2\}\right\}, \end{aligned}$$

where for convenience and brevity the following pairs of labels refer to the same nodes of G': u_5^1 and u_5^2, v_5^1 and v_5^2, $x_{i,i}^j$ and v_i^{3-j} for $i \in \{1,\ldots,4\}$, $j \in \{1,2\}$, $y_{i,4}^1$ and $y_{i,4}^2$ for each $i \in \{1,\ldots,4\}$ (see also Figure 3). The edge set of G' is given by $E(G') = R_1 \cup R_2$, where for $j \in \{1,2\}$

$$\begin{aligned} R_j &= \left\{\{v_i^j, v_{i+1}^j\}, \{u_i^j, u_{i+1}^j\} \mid i \in \{1,\ldots,4\}\right\} \cup \left\{\{v_0^j, v_1^j\}, \{v_0^j, u_0^j\}\right\} \\ &\cup \left\{\{x_{i,i'}^j, x_{i,i'+1}^j\}, \{y_{i,i'}^j, y_{i,i'+1}^j\} \mid i \in \{1,\ldots,4\}, i' \in \{0,\ldots,3\}\right\}. \end{aligned}$$

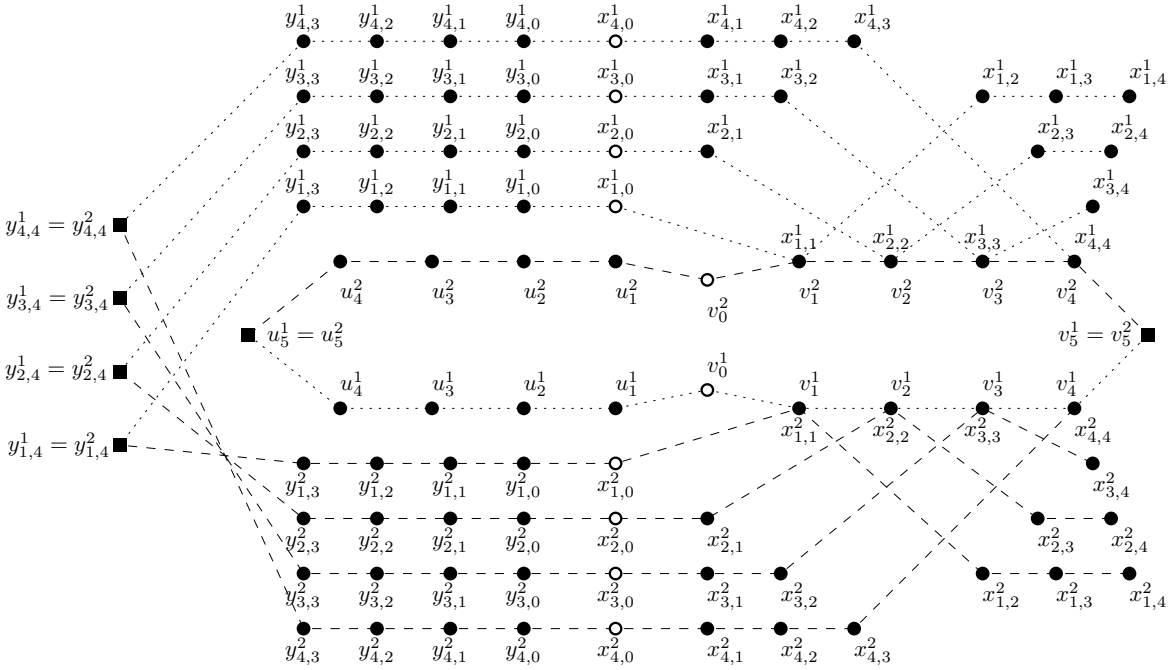

Figure 3: The variable component: squares are exits, dotted edges are in R_1 and dashed edges are in R_2, white nodes denote homebases of agents

For each $j \in \{1,2\}$, we place agents at nodes v_0^j and $x_{i,0}^j$, $i \in \{1,\ldots,4\}$ and we call them agents of *type* j. The set of permitted edges for agents of type j is R_j, $j \in \{1,2\}$. There are six exits in the variable component: u_5^1, v_5^1 and $y_{1,4}^1,\ldots,y_{4,4}^1$ (also denoted equivalently with an upper index 2).

Having described the variable component, we continue the construction of our graph G, taking p disjoint copies of the variable component, where the c-th copy, denoted by G'_c, corresponds to variable $\mathbf{x}_c$, $c \in \{1,\ldots,p\}$. To distinguish the nodes of the c-th copy of the variable component G'_c, $c \in \{1,\ldots,p\}$, we write $x_{i,i'}^j(c)$, $y_{i,i'}^j(c)$, $v_i^j(c)$ and $u_i^j(c)$ in place of $x_{i,i'}^j$, $y_{i,i'}^j$, v_i^j and u_i^j, respectively. In our construction of G, for each $m' \in \{1,\ldots,m\}$ we add two nodes $f_1^{m'}, f_2^{m'}$ that *correspond* to the clause $C_{m'}$. Next, for each $m' \in \{1,\ldots,m\}$ and $t \in \{1,2,3\}$ we follow the rules:

- if the t-th literal in $C_{m'}$ is $\mathbf{x}_c$ for some $c \in \{1,\ldots,p\}$, then pick one node among $x_{1,4}^1(c),\ldots,x_{4,4}^1(c)$ and add to G edges connecting this node with $f_1^{m'}$ and $f_2^{m'}$,
- if the t-th literal in $C_{m'}$ is the negation of a variable $\mathbf{x}_c$, $c \in \{1,\ldots,p\}$, then pick a single node among $x_{1,4}^2(c),\ldots,x_{4,4}^2(c)$ and add to G edges connecting this node with $f_1^{m'}$ and $f_2^{m'}$.

Moreover, in the process of adding the above edges we have to ensure that each node $x^j_{i,4}(c)$, $i \in \{1,\ldots,4\}$, $j \in \{1,2\}$ has at most two incident edges that do not belong to G'_c, $c \in \{1,\ldots,p\}$. Initially we locate the agents in G by placing them in variable components as described above (thus, we have $10p$ agents in total). If $z \in V(G)$ is a homebase of an agent, then such an agent is denoted by $\lambda[z]$. This completes the construction. Clearly, the construction might be computed in polynomial time. □

We have the following observation that highlights the paths that each agent may take in any 5-step evacuation strategy.

Observation 1. *For each $c \in \{1,\ldots,p\}$, if the evacuation completes in 5 steps, then*

(i) *for each $j \in \{1,2\}$, the agent $\lambda[v^j_0(c)]$ traverses path $(v^j_0(c),\ldots,v^j_5(c))$ or the agent traverses $(v^j_0(c),u^j_1(c),\ldots,u^j_5(c))$,*

(ii) *for each $i \in \{1,\ldots,4\}$ and $j \in \{1,2\}$, the agent $\lambda[x^j_{i,0}(c)]$ traverses the path $(x^j_{i,0}(c),y^j_{i,0}(c),\ldots,y^j_{i,4}(c))$ or $(x^j_{i,0}(c),\ldots,x^j_{i,5}(c),f^{m'}_t)$ for some $t \in \{1,2\}$ and $m' \in \{1,\ldots,m\}$.* □

Note that instead of stating the paths that an agent follow, one equivalently may provide the node through which the agent evacuates. This follows from the construction of G, namely from the fact that for each possible exit through which an agent may evacuate the path of length 5 that leads to the exit is unique. The next observation states how the movements of some agents affect the behavior of others.

Observation 2. *For each $c \in \{1,\ldots,p\}$, if the evacuation completes in 5 steps, then*

(i) *if the agent $\lambda[v^1_0(c)]$ evacuates through $v^1_5(c)$ (respectively, $u^1_5(c)$), then the agent $\lambda[v^2_0(c)]$ evacuates through $u^2_5(c)$ (respectively, $v^2_5(c)$),*

(ii) *for each $j \in \{1,2\}$, agent $\lambda[v^j_0(c)]$ evacuates through $v^j_5(c)$ if and only if agents $\lambda[x^j_{1,0}(c)],\ldots,\lambda[x^j_{4,0}(c)]$ evacuate through $y^j_{1,4}(c),\ldots,y^j_{4,4}(c)$.* □

We now prove two main lemmas that essentially imply Theorem 6.

Lemma 4. *Let F and $\mathbf{x}_1,\ldots,\mathbf{x}_p$ be the input to the $(3,4)$-SAT problem and let G be the graph constructed above. If F is satisfiable, then there exists a 5-step evacuation strategy in G.*

Proof. Suppose that a Boolean assignment to variables $\mathbf{x}_1,\ldots,\mathbf{x}_p$ that satisfies F. We prescribe the movements of agents as follows. For each $c \in \{1,\ldots,p\}$, if $\mathbf{x}_c$ is true, then:

- let the agent $\lambda[v^1_0(c)]$ traverse the path $(v^1_0(c),u^1_1(c),\ldots,u^1_5(c))$,
- let the agent $\lambda[v^2_0(c)]$ traverse the path $(v^2_0(c),\ldots,v^2_5(c))$,
- for each $i \in \{1,\ldots,4\}$, let the agent $\lambda[x^1_{i,0}(c)]$ traverse the path $(x^1_{i,0}(c),y^1_{i,0}(c),\ldots,y^1_{i,4}(c))$, and
- for each $i \in \{1,\ldots,4\}$, let the agent $\lambda[x^2_{i,0}(c)]$ traverse the path $(x^2_{i,0}(c),\ldots,x^2_{i,5}(c),z)$, where z is a node that corresponds to some clause.

For each $c \in \{1,\ldots,p\}$, if $\mathbf{x}_c$ is false, then the agents perform the 'opposite' actions, namely:

- let the agent $\lambda[v_0^1(c)]$ traverse the path $(v_0^1(c), \ldots, v_5^1(c))$,
- let the agent $\lambda[v_0^2(c)]$ traverse the path $(v_0^2(c), u_1^2(c), \ldots, u_5^2(c))$,
- for each $i \in \{1, \ldots, 4\}$, let the agent $\lambda[x_{i,0}^1(c)]$ traverse the path $(x_{i,0}^1(c), \ldots, x_{i,5}^1(c), z)$, where z is a node that corresponds to some clause, and
- for each $i \in \{1, \ldots, 4\}$, let the agent $\lambda[x_{i,0}^2(c)]$ traverse the path $(x_{i,0}^2(c), y_{i,0}^2(c), \ldots, y_{i,4}^2(c))$.

This strategy does not specify the nodes z that correspond to clauses and we will argue that they can be selected to ensure that the evacuation strategy is valid. It immediately follows that this strategy performs valid (i.e., collision free) movements of agents within the first four steps. Thus, we analyze the last fifth step of the strategy. Any agent that is about to exit through a node in a variable component can do so and thus it is enough to analyze the agents evacuating through nodes that correspond to clauses. Thus, let $m' \in \{1, \ldots, m\}$ be selected arbitrarily and it is enough to argue that there exist at most two agents that in step 4 occupy nodes adjacent to $f_1^{m'}$ and $f_2^{m'}$. Let $x_{i,4}^j(c)$, $c \in \{1, \ldots, p\}$, $i \in \{1, \ldots, 4\}$, $j \in \{1, 2\}$, be some neighbor of the above nodes. Note that there exist exactly three nodes that belong to variable components and are neighbors of $f_1^{m'}$ and $f_2^{m'}$. Hence, to complete the proof we show that if the literal in $C_{m'}$ that is $\mathbf{x}_c$ or its negation is true, then no agent occupies $x_{i,4}^j(c)$ in step 4. We now assume that the literal is the variable and the proof is analogous for the case when the literal is the negation of $\mathbf{x}_c$. We have that $j = 1$ according to the construction of G. Since $\mathbf{x}_c$ is true by assumption, the agent $\lambda[x_{i,0}^1(c)]$ traverses the path $(x_{i,0}^1(c), y_{i,0}^1(c), \ldots, y_{i,4}^1(c))$ and therefore it is not present at node $x_{i,4}^1$ in step 4. This completes the proof because no other agent can reach the node $x_{i,4}^1$. □

Lemma 5. *Let F and $\mathbf{x}_1, \ldots, \mathbf{x}_p$ be the input to the* $(3,4)$-SAT *problem and let G be the graph constructed above. If there exists a* 5*-step evacuation strategy in G, then F is satisfiable.*

Proof. We define Boolean assignment as follows: $\mathbf{x}_c$ is true if and only if the agent $\lambda[v_0^1(c)]$ evacuates through $u_5^1(c)$ in the evacuation strategy, $c \in \{1, \ldots, p\}$. Take any $m' \in \{1, \ldots, m\}$ and we argue that some literal in $C_{m'}$ is true under the above assignment. The nodes $f_1^{m'}$ and $f_2^{m'}$ have three neighbors in variable components and therefore one of those variable components, say G'_c, has the property that the node of G'_c adjacent to $f_1^{m'}$ and $f_2^{m'}$, let this node be $x_{i,4}^j(c)$, has no agent in step 4. The only agent that can reach $x_{i,4}^j(c)$ is $\lambda[x_{i,0}^j(c)]$. By Observation 1, this agent evacuates through $y_{i,4}^j(c)$ in the evacuation strategy we consider. Thus, by Observations 1 and 2, the agent $\lambda[v_0^j]$ exists through $u_5^j(c)$. By construction of G, $C_{m'}$ has $\mathbf{x}_c$ as a literal if and only if $j = 1$. If $j = 1$, then by definition $\mathbf{x}_c$ is true and consequently $C_{m'}$ is true as required. If $j = 2$, then by Observations 1 and 2, the agent $\lambda[v_0^1(c)]$ evacuates through v_5^1 and therefore $\mathbf{x}_c$ is false, which implies that that $C_{m'}$ is true. This completes the proof. □