Multi-Agent Path Planning in Known Dynamic Environments*

Aniello Murano[1], Giuseppe Perelli[2], Sasha Rubin[1]

[1]Università di Napoli "Federico II" and [2]University of Oxford

Abstract. We consider the problem of planning paths of multiple agents in a dynamic but predictable environment. Typical scenarios are evacuation, reconfiguration, and containment. We present a novel representation of abstract path-planning problems in which the stationary environment is explicitly coded as a graph (called the arena) while the dynamic environment is treated as just another agent. The complexity of planning using this representation is PSPACE-complete. The arena complexity (i.e., the complexity of the planning problem in which the graph is the only input, in particular, the number of agents is fixed) is NP-hard. Thus, we provide structural restrictions that put the arena complexity of the planning problem into PTIME (for any fixed number of agents). The importance of our work is that these structural conditions (and hence the complexity results) do not depend on graph-theoretic properties of the arena (such as clique- or tree-width), but rather on the abilities of the agents.

1 Introduction

The path-planning problem is to find collision-free paths for mobile agents in an environment that may contain obstacles [20, 32, 10, 27]. Obstacles, which may not always be stationary, are *known* if their size, locations, motions, etc. are fixed ahead of planning.

For example, consider a building consisting of rooms, corridors between certain pairs of rooms, and a set of exits. Initially, a fixed number of people are positioned in various rooms, and a flood begins in one of the rooms. At each time step every agent can either stay where it is or move through a corridor to an adjacent room. Suppose the flood spreads radially (i.e., at each time step it reaches all adjacent rooms that are accessible via a corridor). The path planning problem is to exhibit a sequence of actions for the agents that ensures they can reach an exit before the flood traps them.

Other applications are to space exploration, warehouse management, intelligent transportation, assembly or disassembly, and computer games (see [16] and the references therein).

* This work has been partially supported by the FP7 EU project 600958-SHERPA and the ERC Advanced Grant "RACE" (291528) at Oxford. Sasha Rubin is a Marie Curie fellow of the Istituto Nazionale di Alta Matematica.

The AI literature on planning has established that planning is intractable, e.g., propositional STRIPS planning is PSPACE-COMPLETE [2]. To gain deeper insight into what makes planning hard, one must study structural properties of the problem, e.g., [19]:

> For many discrete planning problems that we would like a computer to solve, the state space is enormous (e.g., 10^{100} states). Therefore, substantial effort has been invested in constructing implicit encodings of problems in hopes that the entire state space does not have to be explored by the algorithm to solve the problem.

In this paper we propose an implicit representation ("encoding" in the language above) of path-planning problems of mobile agents in which obstacles are known, and agents should collaborate, rather than compete, to achieve some goal. Know but dynamic environments, such as the flood in the example above, are treated as agents.

Since our representation is exponentially compact, the associated decision problem is PSPACE-complete (Theorems 1 and 2). This is true even for a fixed number of agents, i.e., this high space complexity is not the result of the availability of more and more agents. The *restricted path-planning problem* assumes that all the data is fixed in advance, except for the arena which is the input. We prove that the complexity of the restricted path-planning problem, called the *arena complexity* in the abstract, may be NP-hard (Proposition 1); it is not known if it can be PSPACE-hard. Thus, we describe cases that are solvable in PTIME in which the agent behaviour is restricted (Theorem 3): every agent is either monotone or of bounded-extent. Informally, an agent is monotone if its set of positions can only expand over time (or, only shrink over time). A typical example is the flood: once a room is flooded it stays flooded. An agent is of bounded-extent if it can only occupy a set of positions of bounded size (where the bound does not depend on the size of the arena). A typical example are people: each person in the building can occupy only one room at a time, no matter how large the room is.

1.1 Related Work

Much work in robotics focuses on geometric reasoning, e.g., [32, 30, 18]. Although our environment is discrete (i.e., a finite graph), it may represent a discretisation of the geometric structure of the environment, e.g., a vertex may represent an area not occupied by any of the obstacles. For a discussion of the subtle issues involved in such a translation, see for instance [16].

Standard ways to deal with the fact that planning has, in general, high computational complexity, is to use abstractions and heuristics, see for instance a recent survey on path planning [27]. In contrast, in this paper we isolate computationally tractable but interesting path-planning scenarious.

Standard ways to encode planning problems are logic-based: e.g., the situation calculus is a first-order logic in which the main objects are finite sequences of

actions; and in the set-theoretical representation (such as STRIPS or the multi-agent extension MA-STRIPS [1]) every state is an evaluation of a fixed number of Boolean variables [10]. In contrast, our representation is graph-theoretic and represents the positions of the agents on a graph. Although our planning problems can be expressed in these logical formalisms, this would hide the graphical nature of the problem, see Section 5.

Our representation is related to multi-robot path-planning problems and puzzles [15, 23, 8, 28, 26]. In [15, 28, 26] the goal is to rearrange a group of robots in a given graph, without colliding with each other or with obstacles. The non-optimal version of that problem is in PTIME [15]. We can encode the variation in which more than one agent may move at a time [28], see Example 2. The motion-planning problems and puzzles of [12, 8] are PSPACE-complete. We use one such puzzle to prove a PSPACE-hard lower bound on our path-planning problems, see Theorem 2.

Monotonicity has been used to get PTIME algorithms in the setting of propositional temporal planning [4]. That work studies a propositional representation of planning problems in which the fluents (i.e., literals) are required to be monotone, e.g., once removed a fluent can never be added in any plan of the planning problem. In contrast, the natural translation of our representation into the propositional planning encoding does not preserve monotonicity, see Section 4.

Most of the planning literature, including this paper, focuses on attainment goals of the form "a target configuration can be reached from some initial configuration" [19]. In particular then, we can also model goals of the form "every agent eventually reaches its target vertex in a collision-free way".

Our formalisation and results are inspired by formal methods. Other work in planning with the similar inspirations are an automata-theoretic approach to planning [5], and planning as model checking [24, 11, 31, 25, 21, 13]. These papers also supply centralised planning algorithms, however their representation is based on transition-systems, and hence is not compact, as ours is.

2 Representation of the Path-Planning Problem

We describe how we represent the path-planning problem. Informally, all moving entities (people, floods, fires) are considered to be agents. Agents operate in a finite graph (V, E). At any given time, an agent can occupy a subset V, called its position (e.g., a person occupies a single vertex, and a flood may occupy more than one vertex). One may specify the legal positions $\mathcal{L}$ of the agents, e.g., that agents cannot occupy overlapping vertices. An agent's movements are governed by its mobility relation Δ that relates its current position to its next possible position (the fact that Δ is a relation, rather than a function, models that moves may be nondeterministic). In Section 3 we will discuss how to specify $\mathcal{L}$ and Δ (in particular these objects can be defined algorithmically, or by formulas, e.g., of first-order logic).

Formally, let $k \in \mathbb{N}$ (representing the number the agents). An *arena* is a finite directed graph $\mathcal{A} = (V, E)$. Subsets of V are called *positions*. A *(k-)configuration*

(over $\mathcal{A}$) is an expansion of $\mathcal{A}$ by k many positions, i.e., $\langle V, E, P_1, \cdots, P_k\rangle$ where each $P_i \subseteq V$ is called the *position of agent i (in the configuration)*. Note that an agent can be in more than one vertex, e.g., a flood. A *mobility relation (over $\mathcal{A}$)* is a subset Δ of $2^V \times 2^V$ such that $(X, Y) \in \Delta$ implies $Y \subseteq X \cup E(X)$, where $E(X) := \{v \in V \mid \exists u \in X.(u, v) \in E\}$. The idea is that $(X, Y) \in \Delta$ means that a player can move from position X to position Y but only along edges.

A *(path-)planning domain* is a tuple $\mathcal{D} = \langle \mathcal{A}, \mathcal{L}, \Delta_1, \cdots, \Delta_k\rangle$ where $\mathcal{A} = (V, E)$ is an arena, $\mathcal{L}$ is a set of k-configurations of $\mathcal{A}$, called the *legal configurations*, and each Δ_i is a mobility relation over $\mathcal{A}$. For k-configurations c, d over $\mathcal{A}$ write $c \vdash d$ to mean that d results from c via simultaneous application of Δ_is, formally: for $c = \langle V, E, P_1, \cdots, P_k\rangle$ and $d = \langle V, E, Q_1, \cdots, Q_k\rangle$ define $c \vdash d$ iff $(P_i, Q_i) \in \Delta_i$ for all $i \leq k$. An *execution starting in configuration c* is a finite or infinite sequence π of configurations of $\mathcal{A}$ such that $\pi_1 = c$ and for all i, a) $\pi_i \in \mathcal{L}$, and b) $\pi_i \vdash \pi_{i+1}$.

A *(path) planning instance* is a tuple $\mathcal{P} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G}\rangle$ where $\mathcal{I}$ and $\mathcal{G}$ are sets of k-configurations over $\mathcal{A}$ called the *initial configurations* and *goal configurations*. An execution π of $\mathcal{P}$ is called a *solution* if $\pi_1 \in \mathcal{I}$ and there exists j such that $\pi_j \in \mathcal{G}$.

Example 1 (Evacuation). To model the flood example from the introduction with one person take $k = 2$, i.e., one agent is a person, and the other agent is the flood. Consider an initial configuration of the form $\langle V, E, \{p\}, \{f\}\rangle$, i.e., the person starts in some vertex s, and the source of the flood is in some vertex f. The goal configurations are of the form $\cup_{X \subseteq V} \langle V, E, \{t\}, X\rangle$ for some vertex t. The mobility of the person relates $\{v\}$ to all those $\{w\}$ such that $(v, w) \in E$ or $v = w$, i.e., the person can move to an adjacent position, or stay where it is. The mobility of the flood relates X to the single set $X \cup E(X)$, i.e., the flood expands radially. Thus a typical configuration in an execution is of the form $\langle V, E, \{v\}, F\rangle$ for some $v \in V$ and $F \subseteq V$. Finally, to specify that the person cannot move into a flooded area, define the set of legal configurations $\mathcal{L}$ to be those of the form $\langle V, E, \{v\}, F\rangle$ such that $v \notin F$.

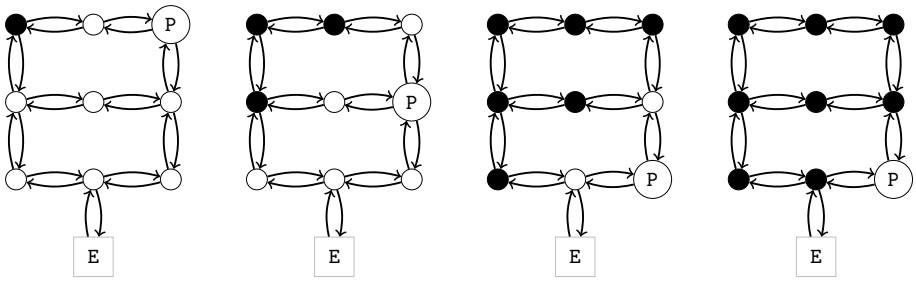

Fig. 1. An evolution of a flood scenario. Player P cannot reach the exit E

In Figure 1 we represent a possible evolution of an instance of the Evacuation scenario. The graph represents three floors of a building, connected by two lifts

located at the sides. Each floor has three connected rooms. Moreover, there is an exit point E at the basement, that is reachable from the central room of the first floor. In the initial configuration, the flood is located in the leftmost top room of the building, while Player P is in the rightmost top. At each step in time, the flood spreads from flooded rooms to the connected unflooded rooms, while P moves to some available room. Unfortunately for P, he gets stuck after three units of time without reaching the exit point. It is not hard to see that, in case the initial position of P is on a side of the second floor (or anywhere in the first floor), he can safely reach the exit point.

Example 2 (Reconfiguration). The abstract path-planning problem requires a set of k mobile robots to re-arrange themselves into a target configuration without colliding with each other or fixed obstacles [28]. This can be modeled as follows. There is one initial configuration, say $\langle V, E, \{v_1\}, \cdots, \{v_k\}\rangle$, and one goal configuration, say $\langle V, E, \{t_1\}, \cdots, \{t_k\}\rangle$. The mobility relation for each agent is the same as the mobility relation for the person in the previous example. Define the set of legal configurations $\mathcal{L}$ to be those of the form $\langle V, E, \{v_1\}, \cdots, \{v_k\}\rangle$ such that $v_i \neq v_j$ for $i \neq j$. This captures the fact that players should not collide.

Example 3 (Containment). Consider a variation of the flooding example in which the goal of the people is to erect barriers in some of the rooms in order to stop the flood from reaching a certain vertex (or cover a set of vertices). This can be modeled by having extra agents that represent barriers. Each barrier is associate with exactly one agent. When a barrier is placed on an unflooded vertex then the flood cannot enter that vertex (this can be coded in the legal-configurations). Agents can carry barriers with them or drop them when they move rooms (this is also expressed in the legal-configurations).

3 Complexity of the Path-Planning Problem

In order to talk about the decision problem for path-planning, we need a way to specify how the environment evolves no matter the size of the arena. For instance, a flood expands radially in any arena. We formalize this as follows.

A *mobility operator* is a function F that maps an arena $\mathcal{A}$ to a mobility relation $\mathsf{F}(\mathcal{A})$ over $\mathcal{A}$. For $k \in \mathbb{N}$, a *k-configuration operator* is a function C that maps an arena $\mathcal{A}$ to a set $\mathsf{C}(\mathcal{A})$ of k-configurations over $\mathcal{A}$. Informally, an operator is *tractable* if there is an efficient algorithm that computes the relation it induces. Formally, a mobility operator F is called *tractable* if there is a polynomial time algorithm that given an arena $\mathcal{A} = \langle V, E\rangle$, and a pair $(X, Y) \in 2^V \times 2^V$, decides whether $(X, Y) \in \mathsf{F}(\mathcal{A})$. Similarly, a configuration operator is *tractable* if there is a polynomial time algorithm that, given an arena $\mathcal{A} = \langle V, E\rangle$ and a configuration $c = \langle V, E, P_1, \cdots, P_k\rangle$ over $\mathcal{A}$ decides whether $c \in \mathsf{C}(\mathcal{A})$. Note that the operators in Examples 1 and 2 are tractable.

We assume, for the rest of this paper, that all operators are tractable.

Remark 1. This tractability assumption implies $\vdash$ is tractable, i.e., there is a PTIME algorithm that given $\mathcal{A}$ and k-configurations c, d over $\mathcal{A}$, decides whether $c \vdash d$.

Observe that fixing the following *planning data*

- $k \in \mathbb{N}$,
- mobility operators F_i (for each $i \leq k$),
- configuration operators $\mathsf{L}, \mathsf{I}, \mathsf{G}$, and
- an arena $\mathcal{A} = \langle V, E \rangle$,

uniquely determines a path-planning domain and a path-planning instance, i.e., the planning domain $\mathcal{D} = \langle \mathcal{A}, \mathsf{L}(\mathcal{A}), \mathsf{F}_1(\mathcal{A}), \cdots, \mathsf{F}_k(\mathcal{A}) \rangle$ and the planning instance $\mathcal{P} = \langle \mathcal{D}, \mathsf{I}(\mathcal{A}), \mathsf{G}(\mathcal{A}) \rangle$. Call this $\mathcal{P}$ the *planning instance induced by the given planning data*, or simply the *induced planning instance*.

Definition 1. *The* path-planning problem *asks, given as input the planning data* $k, \mathsf{F}_1, \cdots, \mathsf{F}_k, \mathsf{L}, \mathsf{I}, \mathsf{G}, \mathcal{A}$*, whether the induced path-planning instance has a solution. The* restricted path-planning problem *fixes* $k, \mathsf{F}_1, \cdots, \mathsf{F}_k, \mathsf{L}, \mathsf{I}, \mathsf{G}$ *and asks, given an arena* $\mathcal{A}$ *as input, whether or not the induced path-planning instance has a solution.*

Remark 2. In order to talk about the complexity of this decision problem (and also of the notion of a tractable operator), we should specify how these objects are coded. We use any natural encoding. E.g., sets V are identified with sets of the form $\{1, 2, \cdots, N\}$; the number of robots k is written in unary; relations (such as $E \subseteq V \times V$ and $P \subseteq V$) are coded by listing their elements; and a tractable operator is coded as the state diagram of a PTIME Turing machine (in Remark 4 we will see how to express operators as formulas rather than machines). The *size* of the planning data is the number of bits of its encoding. Note that the size of an arena $\mathcal{A} = \langle V, E \rangle$ is $O(|V|^2)$, and the size of a k-configuration over $\mathcal{A}$ is $O(|V|^2 + |V|k)$.

Theorem 1. *The path-planning problem can be solved in* PSPACE*, or time exponential in* $|V|$ *and* k*.*

Proof. The number of configurations is $O(2^{|V|k})$, and thus the brute force algorithm takes EXPTIME in $|V|$ and k. However, since each configuration can be written in polynomial space (in the size of V and k), one can search the reachable configuration space using a nondeterministic polynomial-space algorithm. That is, the algorithm stores the current configuration c on its tape, nondeterministically guesses (in PTIME) a legal configuration d (and writes it on its tape), and then verifies that $c \vdash d$ (by Remark 1 the $\vdash$ relation is computable in PTIME). The algorithm begins by guessing (in PTIME) a configuration in $\mathsf{I}(\mathcal{A})$ and proceeds until the current configuration is in $\mathsf{G}(\mathcal{A})$. This algorithm halts if and only if the induced planning problem has a solution. Now use the fact that NPSPACE = PSPACE. $\square$

Corollary 1. *For every* $k, \mathsf{F}_1, \cdots, \mathsf{F}_k, \mathsf{L}, \mathsf{I}, \mathsf{G}$ *the restricted path-planning problem can be solved in* PSPACE*, or time exponential in* $|V|$*.*

Since there are motion-planning problems that are PSPACE-hard (e.g., [12, 8]), the path-planning problem is also PSPACE-hard. For instance, *generalised rush-hour* is a generalisation of a children's puzzle in which the objective is to slide cars in a grid-environment in order for a target car to reach a certain exit co-ordinate. Solving generalised rush-hour is PSPACE-complete [8].

Theorem 2. *The path-planning problem is* PSPACE*-hard.*

Proof. We describe a reduction from generalised rush-hour (GRH) to path-planning problems. Note that although [8] prove PSPACE-hardness when the GRH instances consist of cars (1×2 vehicles) and trucks (1×3 vehicles), it is known that using only cars suffice [29]. An instance of GRH consists of the width w and height h of the grid, a number n of cars, and for each car an orientation (horizontal or vertical) and co-ordinates of the head of each car (x_i, y_i). We assume that the first car is the designated car, and that it has horizontal orientation. Each car can only move, forwards or backwards, in the direction of its orientation. The goal is to move the cars, one at a time, until the head of the designated car reaches the right-hand-side of the grid.[1]

The main problem we have to solve is that in GRH one car moves at a time, while in our path-planning problems agents move concurrently. We solve this problem by treating the set of all cars as a single agent, i.e., let $k = 1$. This introduces the problem that we need to be able to distinguish between different cars. This is solved by placing each car on a disjoint copy of the grid. That is, a GRH configuration is encoded by the arena $w \times h \times n$ grid so that if the head of the ith car is (currently) at co-ordinate (a, b) then: if the car is horizontal then this car is coded by the vertices $a \times b \times i$ and $(a+1) \times b \times i$, and if the car is vertical then this car is coded by the vertices $a \times b \times i$ and $a \times (b+1) \times i$.

More precisely, given an instance of GRH define planning-data as follows. Let $k = 1$, let the arena be the $w \times h \times n$ grid, formally it has $V = [w] \times [h] \times [n]$ and $E = \{((x, y, z), (x', y', z')) : |x - x'| = 1 \text{ xor } |y - y'| = 1 \text{ xor } |z - z'| = 1\}$. We only describe how the operators map grid arenas of the form $a \times b \times c$ for $a, b, c \in \mathbb{N}$ (on other arenas the maps may be arbitrary). The legal configuration operator $\mathcal{L}$ maps the grid arena $a \times b \times c$ to the set of configurations in which there is exactly one car on each of the c-levels; the $\mathcal{G}$ operator maps the grid arena $a \times b \times c$ to the set of legal configurations in which the head of the first car has y-coordinate equal to b; the $\mathcal{I}$ operator maps the grid arena $w \times h \times n$ to the configuration encoding the initial layout of the GRH; the mobility operator maps the grid arena $a \times b \times c$ to the relation over $V = [a] \times [b] \times [c]$ that relates X to Y if the only difference between X and Y is that for some co-ordinate $i \leq c$, every element in $X \cap [a] \times [b] \times [i]$ moves one coordinate forwards in the

[1] The original definition of GRH unnecessarily allows any of the cars to be the target, the exit to be anywhere on the perimeter, and the target car to be horizontal or vertical [8].

direction of the orientation or every element moves one co-ordinate backwards in the direction of orientation. This completes the description of the reduction. It is not hard to see that the GRH has a solution if and only if the planning instance induced by the constructed data has a solution. □

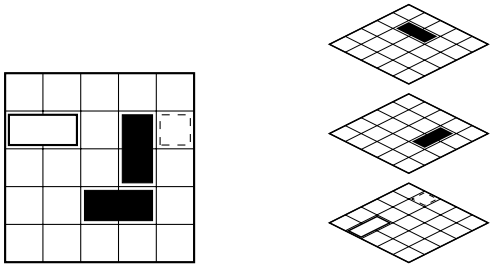

Fig. 2. A representation of the encoding for Generalised Rush Hour (GRH) to path-planning problem. On the left, an instance of GRH. The white rectangle represents the position of the designated car, the black rectangles represent the position of the other cars, the dashed square represents the exit point. On the right, the path-planning representation.

It is not clear if there exists a restricted path-planning problem that is PSPACE-hard. We remark that the proof just given does not work since a reduction would have to map an instance of generalised rush-hour (which includes the initial layout) to an arena $\mathcal{A}$ (which does not include the initial configuration). We do know the following:

Proposition 1. *There exists* $k, \mathsf{F}_1, \cdots, \mathsf{F}_k, \mathsf{L}, \mathsf{I}, \mathsf{G}$ *such that the restricted path-planning problem is* NP*-hard.*

Proof. We reduce from the NP-complete problem that asks if a given vertex colouring of a graph by 3 colours is a proper colouring, i.e., that no two adjacent vertices have the same colour. The idea is to use $k = 3$ agents, the initial configurations are arbitrary colourings (i.e., I maps $\mathcal{A}$ to $\langle \mathcal{A}, P_1, P_2, P_3 \rangle$ such that $P_i \cap P_j = \emptyset$ for $i \neq j$), the mobility relations map $\mathtt{A}$ to the identity relation on 2^V (i.e., agents do not move), every configuration is a legal configuration (i.e., L maps $\mathtt{A}$ to the set of all 3-configurations), and the goal configurations are proper colourings (i.e., G maps $\mathcal{A}$ to $\langle \mathcal{A}, P_1, P_2, P_3 \rangle$ such that $(x, y) \in E$ and $x \in P_i, y \in P_j$ implies $i \neq j$). It is immediate that the reduction that maps $\mathcal{A}$ to the data $k = 3, \mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3, \mathsf{L}, \mathsf{I}, \mathsf{G}$ is computable in PTIME, and that $\mathcal{A}$ is 3-colourable if and only if the constructed restricted path-planning problem has a solution. □

We now identify a general subclass of path-planning problems that are solvable in PTIME. We use the following definitions.

Definition 2. *A mobility relation Δ is called:*

- increasing *if* $(X, Y) \in \Delta \Rightarrow X \subseteq Y$,
- decreasing *if* $(X, Y) \in \Delta \Rightarrow Y \subseteq X$,
- monotone *if it is increasing or decreasing,*
- size-decreasing *if* $(X, Y) \in \Delta \Rightarrow |X| \geq |Y|$.

A mobility operator F *is called* increasing *(resp. decreasing, monotone, size-preserving) if every mobility relation* $\mathsf{F}(\mathcal{A})$ *is increasing (resp. decreasing, monotone, size-preserving).*

A configuration operator C *is called* size-bounded *if there is a constant* $B \in \mathbb{N}$ *such that for every arena* $\mathcal{A}$*, for every* $i \leq k$*, the cardinality of* P_i *in every configuration in* $\mathsf{C}(\mathcal{A})$ *is at most* B.

Note that in Examples 1 and 2, the mobility operators of the players are size-preserving, the mobility operator of the flood is increasing, and the initial-configuration operators are size-bounded (for all the agents, including, although we don't need this fact, the flood).

The following theorem should be contrasted with the fact that Theorem 1 implies that the time-complexity of restricted path-planning problems is exponential in $|V|$.

Theorem 3. *Fix* $B \in \mathbb{N}$*, and consider the planning problem in which one restricts the input* $\langle k, \mathsf{L}, \mathsf{I}, \mathsf{G}, \mathsf{F}_1, \cdots, \mathsf{F}_k \rangle$ *so that* I *is size-bounded (by* B*), and each mobility operator* F_i *is monotone or size-decreasing. The time complexity of the restricted planning problem is polynomial in* $|V|$.

Proof. The number N of reachable configurations is bounded above by a polynomial in $|V|$. Indeed: $N \leq \prod_{i \leq k} m_i |V|^{n_i}$ where $m_i = 1, n_i = 1$ if the mobility operator for agent i is monotone, and $m_i = 2, n_i = B$ if the mobility operator for agent i is size-preserving (since the number of subsets of V of size at most B is bounded above by $2|V|^B$). In particular, the product is $|V|^{O(k)}$, i.e., polynomial in $|V|$ and exponential in k. Now, since the successor configuration relation $\vdash$ is computable in PTIME (see Remark 1) we can build the reachable configuration space in PTIME. Since the initial-configuration and goal-configuration operators are tractable, we can compute the set of initial configurations and the set of goal configurations in PTIME. Now we test if there is a path from some initial configuration to some goal configuration, which can be done in time quadratic in N. □

Remark 3. Thus, the planning instances from Examples 1 and 2 are solvable in PTIME. The algorithm in the proof of Theorem 3 shows that Example 1 is solvable in time $O(|V|^2)$. With k people and one flood the algorithm runs in time $O(|V|^{k+1})$.

Remark 4. The reader who prefers logical specification languages (i.e., FOL) to computational ones (i.e., Turing Machines) may express their operators as formulas. Indeed, since the program-complexity of FOL (i.e., the complexity of model-checking FOL on finite structures in which the formula is fixed and the structure varies) is in PTIME, deduce that every FOL-definable operator is tractable.

Informally, the descriptions of the operators in Examples 1 and 2 can be expressed as formulas of FOL in appropriate signatures.

More precisely, fix unary predicate symbols $P_1, \cdots, P_k, A, B$ and a binary predicate symbol E (we use standard notation, e.g., [7]). We assume we have symbols for equality and set containment with their usual interpretations. A mobility operator F is FOL-*definable* if there is a first-order formula ϕ in the signature $S = (E, A, B)$, such that for every S-structure $\mathcal{M} = (V^{\mathcal{M}}, E^{\mathcal{M}}, A^{\mathcal{M}}, B^{\mathcal{M}})$ we have that $\mathcal{M} \models \phi$ if and only if $(A^{\mathcal{M}}, B^{\mathcal{M}}) \in \mathsf{F}((V^{\mathcal{M}}, E^{\mathcal{M}}))$. For instance, the mobility operator for the flood in Example 1 is definable using a FOL-formula that states that A and B are singletons, say $A = \{a\}, B = \{b\}$, and that $(a, b) \in E$, e.g.,

$$\exists x \in A. \exists y \in B. \forall z. (z \in A \rightarrow x = z) \wedge (z \in B \rightarrow y = z) \wedge (x, y) \in E.$$

Similarly, a configuration operator C is FOL-*definable* if there is a first-order formula ϕ in the signature $(E, P_1, \cdots, P_k)$ such that for every S-structure $\mathcal{M} = (V^{\mathcal{M}}, E^{\mathcal{M}}, P_1^{\mathcal{M}}, \cdots, P_k^{\mathcal{M}})$ we have that $\mathcal{M} \models \phi$ if and only if $\mathcal{M} \in \mathsf{C}((V^{\mathcal{M}}, E^{\mathcal{M}}))$. For instance, a simple variation of the legal configurations in Example 2 is definable by the FOL-formula $\bigwedge_{i \neq j} \neg \exists x. x \in P_i \wedge x \in P_j$.

4 Comparison with other representations

The goal of this section is to analyse the translations of the abstract path-planning problems considered in this paper into standard representations of planning problems, i.e., the set-theoretical representation and the classical representation [10]. We illustrate by encoding the flood problem of Example 1.

A planning problem in the set-theoretical representation is a tuple $\mathcal{S} = \langle P, A, I, G \rangle$ where:

- P is a finite set of *atoms*,
- $\iota \subset P$ is the *initial state*,
- $G \subset P$ is the set of *goal propositions*,
- A is a set of *actions*, each action a is a tuple $\langle pre^+(a), pre^-(a), add(a), del(a) \rangle$ of subsets of P.

A *state* is a subset of P. A *plan* is any sequence (possibly empty sequence $\langle\rangle$) of actions $\pi = \langle a_1, a_2, \cdots, a_N \rangle$. The state $a(s)$ produced by applying action a to state s is defined as follows: if $pre^+(a) \subseteq s$ and $pre^-(a) \cap s = \emptyset$, then $s' := (s \cup add(a)) \setminus del(a)$, and otherwise $s' := s$. The state produced by applying a plan π to a state s is defined inductively: $\langle\rangle(s) := s$ and $\langle \pi, a \rangle (s) := a(\pi.s)$. A plan π is a *solution* if $G \subseteq \pi(\iota)$.

For simplicity of exposition we now encode the flood scenario of Example 1 into the set-theoretical representation. Let $\langle V, E \rangle$ be the underlying graph, $q, r \in V$ be the starting positions of the person and the flood, respectively, and $t \in V$ the exit. Every vertex v of the graph is associated with two atoms p_v, f_v. The meaning of p_v is that the person currently occupies vertex v, and the

meaning of f_v is that vertex v is flooded. The initial state is $\iota = \{p_q, f_r\}$. The goal G is the set $\{t_v\}$. For every triple $\langle v, w, F\rangle \in E \times 2^V$ with $v, w \notin F$, there is an action a defined as follows:

- $pre^+(a) = \{p_v\} \cup \{f_z : z \in F\}$ (if the person is at v and F is flooded...),
- $pre^-(a) = \{f_z : z \notin F\}$ (and nowhere else is flooded...),
- $rem(a) = \{p_v\}$ (then remove the person from v...),
- $add(a) = \{p_w\} \cup \{f_z : \exists x \in F.(x, z) \in E\}$ (and place the person at w and expand theflood).

Then the person can escape the flood if and only if the planning problem $\mathcal{S}$ has a solution.

The translation into the classical representation (i.e., states are ground literals in a first-order relational signature) is similar. In brief, there is a constant v for each vertex, constants p and f for the agents, a relation $at(x, y)$ that says that agent x occupies vertex y, and for each triple $(v, w, F) \in E \times 2^V$ there is an action with precondition $\{at(p, v)\} \cup \{at(f, z) : z \in F\} \cup \{at(f, z) : z \notin F\}$ and effects $\{at(p, w), \neg at(p, v)\} \cup \{at(f, z) : z \in E(F)\}$.

We now present a simple analysis. First, the size of both of the representations is exponential in $|V|$. In contrast, the size of the representation in Example 1 is polynomial in $|V|$. Second, since the actions of the agents are concurrent, none of the actions are monotone, e.g., it is note the case that once a tuple is removed from at (or a variable in the set-theoretic representation is removed) it is not added later. In contrast, our representation neatly separates the abilities of the multiple agents (i.e., some are monotone, others are not) even though the agents behave concurrently. Third, consider the *interaction graph* for the set-theoretic representation [3]: its vertices are the atoms, and there is an edge from atom x to atom y if there is some action a such that $x \in pre^+(a) \cup pre^-(a) \cup rem(a) \cup add(a)$ and $y \in rem(a) \cup add(a)$. The main result in [3] implies that if there is no bound on the size of the strong connected components of the interaction graph then the planning problem (i.e., the existence problem) is not in PTIME unless an established complexity-theoretic assumption fails. Note that the arena $\mathcal{A} = \langle V, E\rangle$ embeds in the interaction graph. Thus there is no bound on the size of the strongly connected components of the interaction graphs, even in the flood example. In contrast, since our framework is tailored for path-planning problems (and not general planning problems), we were able to exhibit PTIME algorithms for such path-planning problems on arbitrary arenas (Theorem 3).

5 Summary and Future Work

We exhibited a natural representation of path-planning problems in which the arena and the positions of the agents in the arena are coded in a natural graph-theoretic way, rather than as an evaluation of Boolean propositions (as in, e.g., the set-theoretical representation). Our formalism can represent static environments (e.g., fixed obstacles are coded by the lack of an edge in the graph) and

dynamic but predictable environments (e.g., flooding). We gave a PSPACE algorithm for solving the planning problem in these settings, which is also solvable in time exponential in the number of robots k and the size of the arena $|V|$.

By restricting the abilities of the agents (to be monotone or size-preserving), we showed that the planning problem can be solved in time polynomial in the size of the arena, i.e., for a fixed number of agents k but a varying arena, these restricted planning problems are PTIME in the size of the arena $|V|$. The same ideas can also be used to get PTIME algorithms for b-bounded piecewise monotone mobility relations, e.g., for $b = 1$ the flood starts spreading, but at some point in time stops spreading and begins to recede. The reason is that the size of the configuration space blows up, for each b-bounded piecewise monotone agent, by a multiplicative factor of $|V|^b$.

The main open question in this work is the exact complexity of restricted planning-problems, i.e., is there a restricted planning-problem that is PSPACE-hard? We know that there is a restricted planning-problem with three agents that is NP-hard (Proposition 1).

This paper opens up many avenues for extending the model and studying the complexity: optimal planning, e.g., there are costs associated with moving that need to be minimised [19] — our paper only deals with feasible planning, i.e., "does there exist a plan"; adversarial agents that are non-deterministic or probabilistic [14, 17, 6, 9, 22] — our paper only covers, implicitly, deterministic adversaries, such as the flood; and more expressive goals, such as those expressible in temporal logics, [5, 24] or quantitative goals — our paper only deals with attainment/reachability goals; imperfect information, e.g., one does not know the exact starting position of the flood, or one does not know the speed of the flood.

References

1. R. I. Brafman and C. Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In J. Rintanen, B. Nebel, J. C. Beck, and E. A. Hansen, editors, *ICAPS*, pages 28–35. AAAI, 2008.
2. T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1):165–204, 1994.
3. H. Chen and O. Giménez. Causal graphs and structurally restricted planning. *J. Comput. Syst. Sci.*, 76(7):579–592, 2010.
4. M. C. Cooper, F. Maris, and P. Régnier. Monotone temporal planning: Tractability, extensions and applications. *J. Artif. Intell. Res. (JAIR)*, 50:447–485, 2014.
5. G. De Giacomo and M. Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In S. Biundo and M. Fox, editors, *Recent Advances in AI Planning, ECP*, volume 1809 of *LNCS*, pages 226–238. Springer, 1999.
6. T. L. Dean, R. Givan, and K. Kim. Solving stochastic planning problems with large state and action spaces. In R. G. Simmons, M. M. Veloso, and S. F. Smith, editors, *AIPS*, pages 102–110. AAAI, 1998.
7. H. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.
8. G. W. Flake and E. B. Baum. Rush hour is pspace-complete, or "why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1-2):895 – 911, 2002.

9. H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2013.
10. M. Ghallab, D. S. Nau, and P. Traverso. *Automated planning - theory and practice*. Elsevier, 2004.
11. F. Giunchiglia and P. Traverso. Planning as model checking. In *Recent Advances in AI Planning*, pages 1–20, 1999.
12. J. Hopcroft, J. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". Technical report, Courant Institute of Mathematical Sciences, New York, 1984.
13. W. Jamroga. Strategic planning through model checking of ATL formulae. In *ICAISC*, volume 3070 of *Lecture Notes in Computer Science*, pages 879–884. Springer, 2004.
14. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.
15. D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *FOCS*, pages 241–250. IEEE, 1984.
16. A. Krontiris, Q. Sajid, and K. Bekris. Towards using discrete multiagent pathfinding to address continuous problems. In *Proc. AAAI Workshop on Multiagent Pathfinding*, 2012.
17. M. Lamiri, X. Xie, A. Dolgui, and F. Grimaud. A stochastic model for operating room planning with elective and emergency demand for surgery. *European Journal of Operational Research*, 185(3):1026–1037, 2008.
18. J.-C. Latombe. *Robot motion planning*, volume 124 of *International Series in Engineering and Computer Science*. Springer, 2012.
19. S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
20. V. J. Lumelsky, A. Stepanov, et al. Dynamic path planning for a mobile automaton with limited information on the environment. *Automatic Control, IEEE Transactions on*, 31(11):1058–1063, 1986.
21. F. Mogavero, A. Murano, and M. Y. Vardi. Relentful strategic reasoning in alternating-time temporal logic. *Journal of Logic and Computation, to appear*, 2014.
22. A. Murano and L. Sorrentino. A game-based model for human-robots interaction. In *Workshop "From Objects to Agents" (WOA)*, volume 1382 of *CEUR Workshop Proceedings*, pages 146–150. CEUR-WS.org, 2015.
23. C. H. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki. Motion planning on a graph (extended abstract). In *FOCS*, pages 511–520. IEEE, 1994.
24. M. Pistore and P. Traverso. Planning as model checking for extended goals in non-deterministic domains. In T. Walsh, editor, *IJCAI*, pages 479–486. IJCAI/AAAI, 2001.
25. M. Pistore and M. Y. Vardi. The planning spectrum - one, two, three, infinity. *J. Artif. Intell. Res. (JAIR)*, 30:101–132, 2007.
26. G. Röger and M. Helmert. Non-optimal multi-agent pathfinding is solved (since 1984). In D. Borrajo, A. Felner, R. E. Korf, M. Likhachev, C. L. López, W. Ruml, and N. R. Sturtevant, editors, *SOCS*. AAAI, 2012.
27. O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau. Path planning: A 2013 survey. In *Industrial Engineering and Systems Management (IESM)*, pages 1–8, Oct 2013.

28. P. Surynek. An application of pebble motion on graphs to abstract multi-robot path planning. In *ICTAI*, pages 151–158. IEEE, 2009.
29. J. Tromp and R. Cilibrasi. Limits of Rush Hour Logic Complexity. *CoRR*, abs/cs/0502068, 2005.
30. J. van den Berg, J. Snoeyink, M. C. Lin, and D. Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and Systems V*, 2009.
31. W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *AAMAS*, pages 1167–1174. ACM, 2002.
32. G. T. Wilfong. Motion planning in the presence of movable obstacles. *Ann. Math. Artif. Intell.*, 3(1):131–150, 1991.