Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

The Simultaneous Maze Solving Problem

Stefan Funke and **André Nusser**
Universität Stuttgart
Institut für Formale Methoden der Informatik
70569 Stuttgart, Germany
{funke,nusser}@fmi.uni-stuttgart.de

Sabine Storandt
Julius-Maximilians-Universität Würzburg
Institut für Informatik
97072 Würzburg, Germany
storandt@informatik.uni-wuerzburg.de

Abstract

A grid maze is a binary matrix where fields containing a 0 are accessible while fields containing a 1 are blocked. A movement sequence consists of relative movements *up, down, left, right* – moving to a blocked field results in non-movement. The simultaneous maze solving problem asks for the shortest movement sequence starting in the upper left corner and visiting the lower right corner for *all* mazes of size $n \times m$ (for which a path from the upper left to the lower right corner exists at all). We present a theoretical problem analysis, including hardness results and a cubic upper bound on the sequence length. In addition, we describe several approaches to practically compute solving sequences and lower bounds despite the high combinatorial complexity of the problem.

Introduction

We consider the problem of finding a short movement sequence that solves a set of $n \times m$ mazes simultaneously. Here, a maze is a binary matrix with 1s indicating obstacles and 0s obstacle-free fields, see Figure 1 for an example. Allowed moves in the maze are *u*p, *d*own, *l*eft, and *r*ight. If a move leads to a blocked field, no movement takes place. The goal is to find a sequence of moves $s \in \{u, d, l, r\}^*$ such that, starting in the upper left corner ('the start') of the maze and following the sequence s, the lower right corner ('the goal') is visited (not necessarily by the last move). In this case, we call s a *solving sequence*. For a single maze, this is an easy problem as the optimal sequence can be computed via a shortest path algorithm in polynomial time interpreting the maze as a graph. But we are interested in solving a set of mazes *simultaneously*. We differentiate two problem versions depending whether we want to solve *all* feasible mazes for given n, m or only a subset. Here we call a maze *feasible* if there exists an obstacle-free path from the start to the goal.

Definition 1 *Given n, m the* All Simultaneous Maze Solving Problem (ASIMASOP) *asks for the shortest solving sequence for all feasible mazes of size $n \times m$.*

Note that for ASIMASOP, we require unary encoding of n and m as otherwise already the output has exponential size in the input.

Definition 2 *Given a set $\mathcal{M}$ of feasible mazes, the* Simultaneous Maze Solving Problem (SIMASOP) *demands to find the shortest solving sequence for all mazes in $\mathcal{M}$.*

If for a solving sequence the final position in all mazes is the goal state, we call it a *perfect solving sequence*. Figure 2 illustrates these concepts for 3×2 mazes.

Finding a solving sequence in practice incurs subproblems interesting on their own, like the enumeration of all feasible $n \times m$ mazes. There are already more than 9 trillion 7×7 mazes, so just verifying a movement sequence becomes a real challenge.

Related Work

The simultaneous maze solving problem can be seen as an instance of conformant planning, where the goal is to synthesize a plan under the assumption that there is no sensing capability during plan execution (Smith and Weld 1998), (Hoffmann and Brafman 2006). In fact, a solution for ASIMASOP could be interpreted as en emergency protocol for a robot whose sensors fail, that allows to reach the exit of a room despite having no knowledge about the number and positions of potential obstacles.

Furthermore, SIMASOP and ASIMASOP can be formulated as instances of the problem of finding a *synchronizing sequence* (Hennine 1964) for a graph. A synchronizing sequence of a labeled graph is a traversal that ends up in one and the same state for every starting state. We can transform our problem to the synchronizing sequence problem by constructing a graph for the union of all mazes in $\mathcal{M}$ and adding one further state q to which all the goal states are connected via all possible labels. The state q then also has a loop edge to itself; hence we stay in this state once entered. A synchronizing sequence for this graph *without the last symbol/move* is a solving sequence for $\mathcal{M}$. As synchronizing sequences allow arbitrary start states (and not only the upper left corner), they might be significantly longer than shortest solving sequences. Furthermore, applying any algorithm for constructing synchronizing sequences seems doomed by the exponential growth of $|\mathcal{M}|$ for ASIMASOP which implies exponential growth of the respective graph.

There is also a close connection of our problem to universal traversal sequences (UTS) (Aleliunas et al. 1979) for labeled d-regular graphs. A d-regular graph is called *labeled*

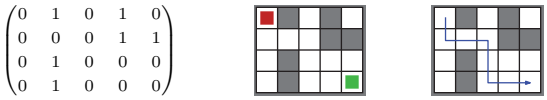

Figure 1: Binary matrix representation (left) of a 4×5 maze (center). Movement sequences $drrddrr, rdrdrdddrdr, drrrdddrrrr$ all correspond to the solving path shown in blue (right).

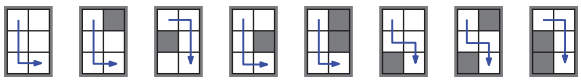

Figure 2: All 8 feasible 3×2 mazes along with their shortest solving sequence of length 5 $ddrdd$, which is also a perfect solving sequence.

if each adjacent edge of a node v is assigned a unique label from $\{0, \ldots, d-1\}$. This labeling does not have to be consistent, i.e., the edge vw might get different labels from v and w. A string $s \in \{0, \ldots, d-1\}^*$ is called a UTS for (N, d) if in *any* connected, N-node, d-regular, labeled graph, starting from an arbitrary node and following s (always choosing the next edge according to the next character in s) leads to a complete traversal of the graph. Finding a UTS is more general than finding a solving sequence. Indeed, every UTS for graphs of size $N = n \cdot m$ and degree $d = 4$ is also a solving sequence for the set of all feasible $n \times m$ mazes. In (Aleliunas et al. 1979) the existence of a UTS of length $\mathcal{O}\left(|V|^3 \log |V|\right)$ is shown. We improve upon this bound for maze solving. [t]

Contribution

This paper provides the following insights on the simultaneous maze solving problem: We first prove SIMASOP to be NP-complete by reduction from CNFSAT. Furthermore, we show that a solving sequence as well as a perfect solving sequence exists for every ASIMASOP instance and provide theoretical lower as well as upper bounds on the sequence length. To allow for practical solutions, we provide an algorithm that explicitly enumerates all feasible mazes for given n, m as well as an algorithm that finds an unsolved maze for a given movement sequence. Based on that we design several practical approaches which find good solving sequences, as well as lower bounds. Table 1 shows what can be achieved with our current implementation. Due to the high combinatorial complexity of ASIMASOP, even for 4×4 the optimal sequence length remains unknown.

Hardness Result

We now prove the decision problem of SIMASOP to be NP-complete. Given a tuple $(\mathcal{M}, k)$, the decision problem asks whether there exists a solving sequence for the set of mazes $\mathcal{M}$ which has length $\leq k$. First, we show SIMASOP $\in$ NP. We can check if there exists a solving sequence shorter than k nondeterministically in polynomial time: we simply guess a sequence of length $\leq k$ and then verify if it really is a

Figure 3: Building blocks for the reduction functions.

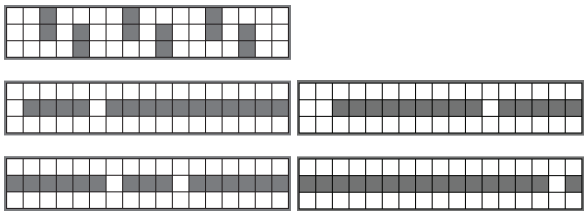

Figure 4: The mazes resulting from $f((A \vee B) \wedge (\neg B \vee C) \wedge (A \vee \neg A \vee \neg C) \wedge (D))$.

solving sequence by naively checking this property for every maze in $\mathcal{M}$. Note that the mazes are part of the input and thus contribute to the input size. This is not the case for ASIMASOP where the input is only n and m, and therefore we cannot prove that ASIMASOP $\in$ NP using the same argument. It remains to show NP-hardness for SIMASOP. This is done by providing a polynomial time reduction from CNFSAT.

Definition 3 (CNFSAT) *Given a Boolean formula F in conjunctive normal form (CNF), i.e., $F = \bigwedge_{i=1}^{c} \bigvee_{j=1}^{c_i} A_{i,j}$. The decision problem CNFSAT asks if there exists a satisfying assignment for F.*

Given a formula in CNF $F = \bigwedge_{i=1}^{c} \bigvee_{j=1}^{c_i} A_{i,j}$, we define the function f that realizes the reduction as $f(F) = (\{M\} \cup \{M_i | 1 \leq i \leq c\}, 9\eta - 6)$ where η is the number of *distinct* variables $X_1, \ldots, X_\eta$ in F, and M and the M_i are mazes as described in the following. The maze M consist of an empty 3×2 block for every variable X_i in F; see Figure 3(a). These blocks appear in order of the index of the variable and in between two of those blocks is one 3×3 block as shown in Figure 3(b). The mazes M_i – which correspond to the clauses C_i of F – consist of a block like shown in Figure 3(c) for every variable X_j, where the green field is *not* blocked if and only if X_j is contained in the clause C_i while the red field is *not* blocked if and only if $\neg X_j$ is contained in the clause C_i. In between those blocks there is a 3×3 block like the one shown in Figure 3(d). The mazes contained in $f(F)$ for the example formula $F = (A \vee B) \wedge (\neg B \vee C) \wedge (A \vee \neg A \vee \neg C) \wedge (D)$ can be seen in Figure 4. It remains to show that f is indeed a reduction.

Lemma 4 *$F \in$ CNFSAT $\Leftrightarrow f(F) \in$ SIMASOP.*

Proof. Let us first show that $F \in$ CNFSAT $\Rightarrow f(F) \in$ SIMASOP. Assume $F \in$ CNFSAT. It follows that there exists an assignment $\mathcal{A}$ under which F evaluates to *true*. From the assignment $\mathcal{A}$ we construct a solving sequence s by iterating over the assignment of the variables in order. If a variable X_i is set to *true*, we append $s_i = ddr$, else $s_i = rdd$. In between we always append $rruurr$. Note that this constructs a sequence of length $9\eta - 6$, which was chosen as k of

$n\backslash m$	2				3				4				5				6			
	l_t	l_p	u_p	u_t	l_t	l_p	u_p	u_t	l_t	l_p	u_p	u_t	l_t	l_p	u_p	u_t	l_t	l_p	u_p	u_t
2	2	3	3	480	3	5	5	1680	4	7	7	4032	7	11	11	7920	8	13	13	13728
3					4	11	11	5760	5	17	17	13728	10	26	67	26880	11	27	111	46512
4									6	27	29	32640	13	28	222	63840	14	29	434	110400
5													16	29	551	124800	19	30	1520	215760
6																	20	31	3728	372960

Table 1: The entries l_t, l_p, u_p, u_t denote the theoretical (l_t) and practical (l_p) lower bounds, as well as the practical (u_p) and theoretical (u_t) upper bounds for ASIMASOP on $i \times j$ mazes. For example, the entries in row 4, column 4 denote that the a priori lower bound for the length of solving sequences for all 4×4 mazes is 6, whereas we could prove algorithmically that such a sequence must have length at least 27. We could also algorithmically find a solving sequence of length 29, whereas our analytic cubic upper bound is 32640. If $l_p = u_p$, the optimal solution is known. But this only applies up to 4×3.

the decision problem. Also note that s is a solving sequence for M. It remains to show that it is a solving sequence for each M_i. For this, consider the variable X_j with the smallest index that makes C_i evaluate to *true*. Before we execute the sequence s_j appended to s for X_j, we are in the first row of M_i. By construction, we are in the third row of M_i after executing s_j. Note that no up movement can bring us back to the first row. As – again by construction – no right movement of s is blocked in M_i, we end up in the goal state after executing s in M_i.

For the second half of the proof we have to show that $f(F) \in$ SIMASOP $\Rightarrow F \in$ CNFSAT. For this assume $f(F) \in$ SIMASOP and let $(\mathcal{M}, k) := f(F)$. Furthermore, let $s, |s| \leq k$ be the sequence that solves $\mathcal{M}$. By construction of the reduction $|s| = k$, which is the length of the shortest solving sequence of M. Thus, no right movement of s is blocked in M or any of the M_i and we are therefore always in the same column in all of the mazes during the execution of s. As s solves all the mazes in $\mathcal{M}$, we have to cross the second row at some point of the traversal. The crossings of the middle row imply an assignment $\mathcal{A}$ that satisfies F. If we cross at the green field of Figure 3(c), then we set $X_j = 1$, else we set $X_j = 0$. The variable that corresponds to the block where the middle row of M_i is crossed satisfies the clause C_i. As s is a solving sequence for $\mathcal{M}$, all middle rows are crossed and thus $\mathcal{A}$ satisfies F. ■

Existence of a Solving Sequence

Next, we are going to show that there indeed exists a solving sequence for any ASIMASOP (and therefore also any SIMASOP) instance and provide a simple algorithm to find such a sequence. In addition, we prove that also a perfect solving sequence always exists. However, as this proof leverages the probabilistic method, it is not constructive.

Solve in Order To find a solving sequence for a set of mazes $\mathcal{M}$, we put them into an arbitrary ordering and then solve one after another while appending to the already existing sequence. This means: we solve the first maze, then execute this sequence in the second maze, then solve the second maze from the position we end up in, then we execute the resulting sequence in the third maze, solve the third maze, and so on. This algorithm always produces a feasible solving sequence, but the order in which the mazes are considered heavily influences the sequence length in the end.

Perfect Solving Sequence Next, we prove an even stronger result – but unfortunately not in a constructive way.

Theorem 5 *For given n, m, there exists a perfect solving sequence for all feasible mazes of size $n \times m$.*

Proof. For $n < 2$ or $m < 2$ the claim is obviously true. The idea of the proof is to create a random sequence according to the probabilities p_u, p_d, p_l, p_r which are the probabilities of moving up, down, left, and right, respectively. This defines a Markov chain. By choosing the length of the random sequence large enough, we can show – because of convergence to a stationary distribution – that the probability of being in the goal state in all mazes $M \in \mathcal{M}$ is larger than $\frac{3}{4}$. Thus, there exists a sequence with the property of being a perfect solving sequence.

Let the probabilities be given as $p_u = p_l = c$, $p_d = p_r = \frac{1}{2} - c$, $c = \left(\frac{1}{4}\right)^{nm} \leq \frac{1}{4}$. We claim that the stationary distribution of *every* Markov chain corresponding to a maze in $\mathcal{M}$ is given by $p_{i,j} \propto \left(\frac{\frac{1}{2}-c}{c}\right)^{i+j}$ where $p_{i,j}$ denotes the probability of being in position (i, j) if the field is reachable from the starting field. Note that this distribution does not depend on any structure of the maze except the reachability of the fields from the starting state – which is incorporated in the proportionality.

We now show that this is indeed the stationary distribution. The probabilities obviously sum up to 1 . To show that the probabilities stay the same after one step, we show at first that it does not matter for the calculation if a field is next to a wall or a free field. And secondly, that the sum evaluates to the claimed value. The following equation shows that the sum's value does not change if there is a wall or a free field above or to the left. The "below" and "left" case is analog and thus omitted.

$$p_{i,j-1} \cdot p_r = p_{i-1,j} \cdot p_d = \left(\frac{\frac{1}{2} - c}{c}\right)^{i+j-1} \cdot \left(\frac{1}{2} - c\right)$$

$$= \left(\frac{\frac{1}{2} - c}{c}\right)^{i+j} \cdot c = p_{i,j} \cdot p_u = p_{i,j} \cdot p_l$$

Now we show the stationarity. We use $v_{i,j}$ to denote the node of the Markov chain corresponding to position (i,j).

$$\sum_{v_{k,l}\in V} p_{k,l}p((v_{k,l},v_{i,j}))$$
$$= 2\left(\frac{\frac{1}{2}-c}{c}\right)^{i+j}\cdot c + 2\left(\frac{\frac{1}{2}-c}{c}\right)^{i+j}\cdot\left(\frac{1}{2}-c\right)$$
$$= \left(2c+2\left(\frac{1}{2}-c\right)\right)\cdot\left(\frac{\frac{1}{2}-c}{c}\right)^{i+j} = p_{i,j}$$

From the fact that the probability of being in the goal state is not equal to 1, we can derive that: in the stationary distribution the (non-proportional) probability to be in a certain state that is not the goal state is less than $4c$:

$$\left(\frac{\frac{1}{2}-c}{c}\right)^{n+m} < 1 \Leftrightarrow \left(\frac{\frac{1}{2}-c}{c}\right)^{n+m-1} < \frac{c}{\frac{1}{2}-c}$$

and $$\frac{c}{\frac{1}{2}-c} \le \frac{c}{\frac{1}{2}-\frac{1}{4}} \le 4c$$

As the defined Markov chains are not periodic, every initial distribution converges to the stationary distribution. Thus, for a long enough sequence it holds for every maze that

$$P(\text{not in goal state}) < 4c(nm-1) = 4\left(\frac{1}{4}\right)^{nm}(nm-1)$$

where $(nm-1)$ is the maximal number of reachable fields other than the goal. The equality results from plugging in c. It then follows with Boole's inequality:

$$P(\text{not all in goal state}) \le \sum_{M\in\mathcal{M}} 4\left(\frac{1}{4}\right)^{nm}(nm-1)$$
$$\le 2^{nm-2}\cdot 4\left(\frac{1}{4}\right)^{nm}(nm-1) \le \frac{1}{4}$$

The second inequality holds as there are less than 2^{nm-2} feasible mazes of size $n\times m$. It follows that $P(\text{all in goal state}) = 1 - P(\text{not all in goal state}) \ge 1-\frac{1}{4} = \frac{3}{4} > 0$ ∎

Lower and Upper Bounds on the Sequence Length

Next, we show that the length of an optimal solving sequence can always be bounded a priori. We provide a lower bound linear in nm and an upper bound which is cubic in nm, also proving ASIMASOP ∈ PSPACE.

Lower Bounds For ASIMASOP we can easily see that the length of solving sequences for smaller sizes are lower bounds for the length of solving sequences for larger sizes (i.e., the solving sequence for $n\times m$ is shorter than for $n+1\times m$). This holds as the smaller mazes are contained as sub-mazes in the larger mazes and so already a subset of the larger mazes implies this lower bound. The relation is strict as we have to take at least one additional step to reach the new goal field. Another rather naive lower bound is the length of the longest shortest path of a maze in $\mathcal{M}$. For ASIMASOP, we can easily give a lower bound on that number. By blocking the right fields we can always create a zigzag path as shortest solving sequence which goes all to the bottom, two steps to the right, all to the top, two steps to the right, and so on. The exact length of such a zigzag path is $\left(2\left\lfloor\frac{m-1}{4}\right\rfloor+1\right)(n+1)+((m-1 \mod 4)-2)$; so, approximately $\frac{nm}{2}$. Note that we might obtain a better lower bound if we construct a "horizontal zigzag path". To get its length, simply swap n and m in the formula.

Upper Bounds We already showed that UTS is a more general concept of a solving sequence for ASIMASOP. In (Aleliunas et al. 1979) the authors prove that there always exists a UTS with length $\mathcal{O}((nm)^3\log(nm))$, using our notation. Simply calculating the shortest solving sequence by brute-force is therefore possible in polynomial space. We only have to hold the current sequence and the maze we are currently checking in memory. This implies ASIMASOP ∈ PSPACE.

We now improve the upper bound on the solving sequence length by explicitly taking the maze structure into account.

Theorem 6 *For every set of feasible $n\times m$ mazes there exists a solving sequence of length $\mathcal{O}((nm)^3)$.*

Before we conduct the main proof, we state an important result about random walks that is needed for the proof. The proof of this claim is included in (Aleliunas et al. 1979) as Theorem 4 and in (Motwani and Raghavan 1995) as Theorem 6.8. We therefore omit it here.

Lemma 7 *Let $G=(V,E), |V|=n, |E|=m$ be an undirected, connected graph and let $T(v,\cdot)$ be the cover time of G starting from $v\in V$, i.e., the expected number of edge traversals by a random walk starting in v until all vertices in G have been visited. Then $T(v,\cdot)\le 2m(n-1)$.*

Now we give the proof for Theorem 6.

Proof. Let s be a random sequence with $|s| = 2\cdot 4nm(nm-1)(nm+1) \in \mathcal{O}\left((nm)^3\right)$. The moves of s – which can be interpreted as random variables – are chosen independently and with all moves being equally likely. Let $\mathcal{M}_{n,m}$ be the set of all feasible $n\times m$ mazes, and let $X_M, M\in\mathcal{M}_{n,m}$ be a random variable such that:

$$X_M = \begin{cases} 0, & \text{if } s \text{ is a solving sequence of } M \\ 1, & \text{otherwise} \end{cases}$$

Additionally, let $Y := \sum_{M\in\mathcal{M}_{n,m}} X_M$. The sequence s is a solving sequence for $\mathcal{M}_{n,m}$ iff $Y=0$. Therefore, if we can show that $E[Y]<1$, it follows that there exists a solving sequence of length $|s|$. We claim that $E[X_M]\le 2^{-(nm+1)}$. With $|\mathcal{M}_{n,m}|\le 2^{nm}$ it then follows that: $E[Y] = E\left[\sum_{M\in\mathcal{M}_{n,m}} X_M\right] = \sum_{M\in\mathcal{M}_{n,m}} E[X_M] \le 2^{nm}\cdot 2^{-(nm+1)} < 1$. Therefore, it only remains to show that indeed $E[X_M]\le 2^{-(nm+1)}$. For this, let us consider s as a concatenation of $nm+1$ random sequences $s_1,\ldots,s_{nm+1}$, each being of length $2\cdot 4nm(nm-1)$. Let Z be a random variable that equals the number of transitions we need to visit all reachable fields in a maze, starting from an *arbitrary field*, when executing a random sequence. Using Lemma 7,

n	1	2	3	4	5	6
#feasible mazes	1	3	51	3,828	1,225,194	1,636,193,228
n	7					
#feasible mazes	9,009,490,924,794					

Table 2: Number of feasible mazes of size $n \times n$ according to (The On-Line Encyclopedia of Integer Sequences).

we know that $E[Z] \leq 2 \cdot 2nm(nm-1)$ (as the graph is undirected). With Markov's inequality we then get:

$$P(Z \geq 2 \cdot 4nm(nm-1)) \leq \frac{E[Z]}{2 \cdot 4nm(nm-1)}$$
$$\leq \frac{2 \cdot 2nm(nm-1)}{2 \cdot 4nm(nm-1)} \leq \frac{1}{2}$$

In other words, the probability that during the execution of $s_i, 1 \leq i \leq nm+1$ we do not visit the goal field – starting on an arbitrary field – is at most $\frac{1}{2}$. Therefore, the probability that we do not visit the goal field when executing $s = s_1 \dots s_{nm+1}$ is: $P(X_M = 1) \leq \left(\frac{1}{2}\right)^{nm+1} = 2^{-(nm+1)}$ By plugging in the definition of the expected value we get: $E[X_M] = P(X_M = 1) \leq 2^{-(nm+1)}$ ∎

Enumeration of Feasible Mazes

Number of Feasible Mazes

The exact number of feasible mazes of size $n \times n$ for n up to 7 is part of the OEIS (The On-Line Encyclopedia of Integer Sequences), see Table 2 for the results. While the numbers are listed there, no explicit formula or algorithm to calculate them is given. We will calculate the number of feasible mazes in the following by enumerating them.

Practical Enumeration

To calculate the exact number of feasible mazes, we enumerate them via their *solving right-hand path*. The solving right-hand path of a maze is the path we get when solving the maze while always keeping a wall at our right-hand side, starting in the upper-left corner. This technique is guaranteed to produce a solving sequence for a maze as long as the maze is feasible. In Figure 5 the blue trajectories are all solving right-hand paths.

Note that every feasible maze has exactly one solving right-hand path. However, some feasible mazes have the same solving right-hand path. So, instead of enumerating the mazes themselves, we enumerate all the possible solving right-hand paths. Then we calculate the number of feasible mazes for every solving right-hand path and add them up. To get all possible solving right-hand paths we use a simple tree search. Every node of the tree contains a maze and a position in this maze. The root of the tree contains an empty maze with the current position being the start field. For every possible move in the maze of the parent node a child node is created. A child is pruned if the move contradicts a solving right-hand path. Note that by performing the moves, we might have to set fields to be blocked or free for the right-hand path property to be preserved. A solving right-hand path is found when the goal field is reached. How do we now calculate the number of mazes for a given solving right-hand path? This is done by counting the fields in the maze which can still be arbitrarily set. There are two types of fields which cannot be arbitrarily set: the fields on the solving right-hand path (they have to be free) and the fields that define the solving right-hand path (they have to be blocked). A blocked field is said to define a solving right-hand path if the solving right-hand path changes if the field is not blocked. Let the number of fields that can still be arbitrarily set be a. Then the number of different mazes with a certain solving right-hand path is 2^a, i.e. all possible assignments of those fields. See Figure 5 for an example.

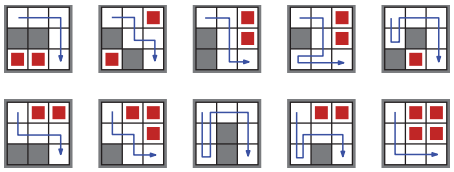

Figure 5: All possible right hand paths in a 3×3 maze. The red-marked fields can be set arbitrarily as they do not affect the right hand path. Thus, the number of feasible 3×3 mazes is: $2^2 + 2^2 + 2^2 + 2^2 + 2^1 + 2^2 + 2^3 + 2^0 + 2^2 + 2^4 = 51$.

Finding Unsolved Mazes

There are several reasons why we need algorithms that find unsolved mazes given a sequence s. First, we can use those algorithms to check if s is a solving sequence. Secondly, we will use this as a black box for our practical algorithms.

The general idea is to perform a depth first search where the depth of the tree corresponds to the position in s. The children of a node depend on whether we are blocked in this step or not. Thus, in every node we have a position in a certain maze with fields that are either blocked, free or unknown. The maze of the root node has two free fields – start and goal – and all the others are unknown. Note that often a child node will not be created because it contradicts the maze of its parent or because it is not feasible. This is important as it prunes the tree and therefore greatly reduces the runtime of the search. See Figure 6 for an illustration.

Computing Solving Sequences in Practice

Exact Algorithms

Brute Force. The naive way to calculate a shortest solving sequence is to try for every possible sequence, in ascending order of their length, if it is a solving sequence for the given set of mazes. The runtime is in $\mathcal{O}\left(4^k \cdot |\mathcal{M}|\right)$ where k is the length of the shortest solving sequence of $\mathcal{M}$. We have 4 different moves to choose from for all sequences up to the length of k, and for every sequence we have to check if it solves all the mazes in $\mathcal{M}$. Note that if $\mathcal{M}$ is not explicitly given as in ASIMASOP and the implicitly used $\mathcal{M}$ is too large to compute explicitly, we have to use an algorithm to find an unsolved maze.

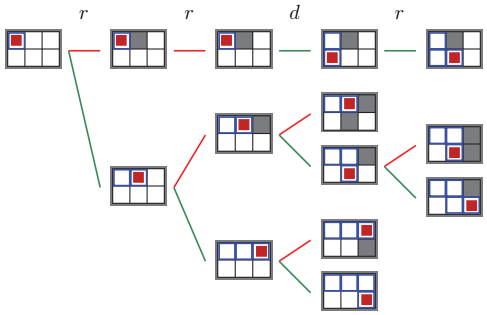

Figure 6: Tree for 3×2 mazes and $s = rrdr$. There are two types of children: The ones connected with a red edge, which are those where the move is blocked and the ones connected with a green edge in the other case. The current position is marked with a red square. Blocked fields (gray) have to be blocked. The mandatory free fields are marked with blue color. The tree is constructed by a depth first search where first "green children" are explored. In two of the leaves we stop exploring because we found a solved maze; in three of the leaves we stop because the maze is not solvable anymore; and in the upper leaf we find an unsolved maze, which is then returned by the algorithm.

A*. To improve the runtime, we can use Dijkstra's algorithm (Dijkstra 1959) or A^* (Hart, Nilsson, and Raphael 1968) to find a shortest solving sequence. The set of nodes of the graph that the algorithm is working on is the power set of states of the single mazes. The edges are possible transitions between those states. The runtime on a set of mazes $\mathcal{M}$ of size $n \times m$ is in $\mathcal{O}\left((nm)^{|\mathcal{M}|} \cdot |\mathcal{M}| \log(nm)\right)$. For A^*, we use the longest shortest path *of all mazes* to the goal state as heuristic. For A^* not to expand a node more than once, the used heuristic has to be consistent. A heuristic h is consistent iff $h(v) \leq c(v, w) + h(w)$ and $h(\text{goal state}) = 0$, where c is the transition cost function. If we are in the goal state, then the longest shortest path is 0, i.e., $h(\text{goal state}) = 0$. In our scenario, we have $c(v, w) \in \{0, 1\}$. If $c(v, w) = 0$, we stay in the same state ($v = w$) and the inequality holds. If $c(v, w) = 1$, we know that we took one step and the heuristic can decrease at most by one. Thus, the inequality holds. Consistency follows.

Exact Solving Sequence (ESS). ESS iteratively builds a set of mazes and computes a shortest solving sequence for this set. In every round a new unsolved maze is included in the set. By construction, the length of the solving sequence stays the same or increases in every round. Additionally, we know that we found a shortest solving sequence if there is no unsolved maze left. To compute a shortest solving sequence for a set of mazes, we can use A^*. To find an unsolved maze, we can use the already presented algorithm.

Heuristic Solutions

Random Sequence (RS). A rather naive way of finding a solving sequence is to create a long random sequence (mimicking the probabilistic argument of the proof of Theorem 6) and then check if it solves all mazes. The probability for it being a solving sequence increases with its length. Interestingly, this method produces good results as we show in our experiments.

Solve in Order (SO). We gave a constructive proof for the existence of a solving sequence, which implies a heuristic algorithm. By choosing different orderings, we can improve the approximation in practice. This method requires that it is feasible w.r.t. runtime to iterate through all the mazes.

Iteratively Append to Sequence (IAS). To make the SO algorithm work, we just need one unsolved maze in every round. If the explicit construction of all mazes is too expensive, we use our described algorithm to find an unsolved maze as a remedy. So, we iteratively compute an unsolved maze and then append moves to the already existing sequence to also make it a solving sequence for this maze.

Greedy Lookahead (GL). The ESS algorithm always computes a shortest solving sequence for a set of mazes. However, instead of computing an exact shortest solving sequence with this function, we can also just use an approximation. We compute an approximate shortest solving sequence by greedily choosing the best sequence (according to some value function) out of a sequence of lookahead sequences $\mathcal{L}$. In our experiments we set $\mathcal{L}$ to be the set of all sequences of length 3, and the value function to be the sum of the squared distances to the goal in every maze. This change accelerates the algorithm significantly while slightly impairing the length of the returned solving sequence.

All of the above methods can be combined with a simple minimization postprocessing where moves are deleted from the solving sequence while maintaining feasibility.

Practical Lower Bounds

Above, we considered heuristics for finding good solving sequences in practice. The length of the resulting sequence constitutes an *upper* bound on the length of the shortest solving sequence. In the algorithms that compute an exact shortest solving sequence, we can use any intermediate result as a *lower bound*.

Experimental Results

We implemented all discussed approaches in Python. Experiments were conducted on an Intel Core i7-4510U CPU and 12GB of RAM.

Enumeration. Our explicit enumeration algorithm worked well for small n, m. The 30,754,544 feasible mazes for 5×6 (induced by 152,668 right-hand paths) could be enumerated in less than a second. For 7×7 (the largest known value in the On-Line Encyclopedia of Integer Sequences), it took about 5 minutes to list the more than 83 billion mazes by enumerating the respective right-hand paths. The largest values that we could handle was 7×8, which took about 224 minutes and resulted in 939,356,338,522,456 (over 900 trillion) mazes. Given better hardware and more computation time, we think that 8×8 should be possible with our algorithm; but for larger values, new insights might be required.

Finding Unsolved Mazes. The tree based method to find an unsolved maze outperformed the naive baseline (parsing

n	RS	IAS	GL	SO
1	< 0.1	< 0.1	< 0.1	0.3
2	< 0.1	< 0.1	< 0.1	0.9
3	2.7	< 0.1	< 0.1	10.9
4	170.7	0.3	8.7	842.3

Table 3: Runtimes (in seconds) for ASIMASOP with size $n \times n$. The times are without minimization, except for RS where minimization is inherent. As RS and SO are randomized, we executed them 10 and 20,000 times, respectively.

n	RS	IAS	GL	SO
2	250 (3)	3 (3)	3 (3)	3 (3)
3	250 (14)	20 (17)	15 (14)	12 (11)
4	250 (51)	109 (53)	74 (53)	50 (29)
5	–	551	–	–
6	–	3728	–	–

Table 4: Approximate shortest solving sequence length of the different algorithms for ASIMASOP with size $n \times n$. In brackets we note the length of the minimized sequence.

through all mazes) by far. The gap grows with increasing maze size and number of mazes. Finding an unsolved maze of size 8×8 for a random sequence of length 1000 took only a few minutes, while explicit enumeration was infeasible.

Exact Solutions. Brute force took several minutes on 3×3 and was already infeasible for 3×4. A^* could solve 3×4 in 99 seconds, ESS in about 5 seconds. For 4×4, unfortunately no exact solution could be computed by any of the approaches.

Heuristic Solutions. With our proposed heuristics we could compute solving sequences up to 6×6. In fact, this was possible using IAS, which depends on our fast routine to find an unsolved maze; but nevertheless it already took several days for 6×6 (while 5×5 took a few hours). The other approaches only produced results up to 4×4. Table 3 presents the runtimes of all heuristics (RS, IAS, GL and SO) for ASIMASOP and Table 4 shows a comparison of the produced sequence lengths. Using SO leads to the shortest sequences. But thousands of runs with randomly permuted maze orders were necessary to obtain these results. Minimization costs (for all heuristics) are a few seconds for 3×3 but already a few minutes for 4×4. Hence SO with minimization quickly becomes expensive. As it relies on explicit enumeration of the mazes, large instances cannot be solved.

Lower Bounds. The best practical lower bounds we found are reported in Table 1. They are all due to the ESS algorithm which easily outperformed all other suggested approaches w.r.t. runtime and solution quality. With the help of these lower bounds we can estimate the quality of our found solutions. For 4×4, the gap is very small. We can prove that the optimal solution length is between 27 and 29. But already for 4×5, our best solution is about a factor of 8 larger than the lower bound, for 6×6 the factor is over 100. So either better heuristics or better lower bounds (or both) are necessary to narrow down this gap.

Conclusions and Future Work

We considered the problem of computing short solving sequences for sets of $n \times m$ mazes. Apart from establishing NP-hardness for SIMASOP, we showed a linear lower and a probabilistic cubic upper bound (in the size of the mazes) for the length of solving sequences both in the SIMASOP and ASIMASOP case. The complexity status of ASIMASOP is unclear, though. While we showed ASIMASOP to be in PSPACE, a proof for PSPACE-hardness is unknown. From a practical point of view, we were able to compute exact solutions for mazes up to size 4×3, and heuristic solutions up to 6×6, solving interesting subproblems for moderate instance sizes along the way. A natural goal is to narrow the gap between upper and lower bounds for larger problem instances. Parallelization of our approaches could help. However, for much larger instances, additional insights in the structure of the problem are probably necessary.

Acknowledgment The investigation of this problem as well as some algorithms and proofs were inspired by discussions on StackOverflow (Stack Overflow) and XKCD (XKCD Forums).

References

Aleliunas, R.; Karp, R. M.; Lipton, R. J.; Lovasz, L.; and Rackoff, C. 1979. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Ann. Symp. on Foundations of Computer Science*, 218–223.

Dijkstra, E. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Hennine, F. C. 1964. Fault detecting experiments for sequential circuits. In *Proc. 5th Ann. Symp. on Switching Circuit Theory and Logical Design*, 95–110.

Hoffmann, J., and Brafman, R. I. 2006. Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence* 170(6):507–541.

Motwani, R., and Raghavan, P. 1995. *Randomized algorithms*. Cambridge University Press.

Smith, D. E., and Weld, D. S. 1998. Conformant graphplan. In *AAAI/IAAI*, 889–896.

Stack Overflow. Solve all 4×4 mazes simultaneously with least moves. Original URL: http://stackoverflow.com/questions/26910401.

The On-Line Encyclopedia of Integer Sequences. Number of $n \times n$ binary arrays with path of adjacent 1's from upper right corner to lower left corner. Original URL: https://oeis.org/A069343.

XKCD Forums. Solve all of the mazes [solutions]. Original URL: http://forums.xkcd.com/viewtopic.php?f=3&t=99534.