Solving Moving-Blocks Problems

André G. Pereira[1], **Luciana S. Buriol**[1] **(Advisor), Marcus Ritt**[1] **(Coadvisor)**

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul – Brazil

`{agpereira,buriol,marcus.ritt}@inf.ufrgs.br`

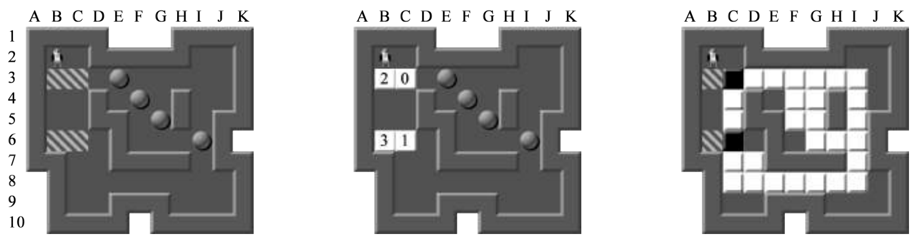
Abstract. *Moving-blocks problems are extremely hard to solve and a representative abstraction of many applications. Despite their importance, the known computational complexity results are limited to few versions of these problems. In addition, there are no effective methods to optimally solve them. We address both of these issues. This thesis proves the* PSPACE*-completeness of many versions of moving-blocks problems. Moreover, we propose new methods to optimally solve these problems based on heuristic search with admissible heuristic functions and tie-breaking strategies. Our methods advance the state of the art, create new lines of research and improve the results of applications.*

1. Motivation

Planning is a general problem-solving model and has been an active research area of *artificial intelligence* for more than half a century. A planning problem is defined by an initial state, a set of goal states and a set of actions. A solution is a sequence of actions that transforms the initial state into some goal state. *Transportation* problems are the most common version of planning problems. They share four fundamental characteristics: a set of connected locations, a set of movable objects, a set of goal locations, and a solution moves (a subset of) movable objects to goal locations. Many transportation problems are motivated by applications such as distribution chain planning [McDermott 2000], airport ground traffic control [Hoffmann et al. 2006], and space exploration [Long and Fox 2003].

Moving-blocks problems are prototypical transportation problems. In these problems, movable blocks are placed on a grid-square maze defined by immovable blocks. There is a distinguished block, called the man, that is the only block that can be moved directly and indirectly moves other blocks. Different moving-blocks problems are defined by the available set of actions and the definition of the set of goal states. The available actions are `Push`, `Pull`, or both `Push` and `Pull` denoted by `PushPull`. The man may be able to move only a fixed number of blocks (e.g. 1, k for some $k > 1$) or an unlimited number of blocks (denoted by $*$) at once. There are problems variants in which a moved block slides until it hits an obstacle, denoted by the repetition of the available action (e.g. `PushPush`). The two most common definitions of the set of goal states are *Storage* where each block must be placed at a distinct goal location and *Path* where there is only one goal location which the man must reach.

Sokoban (`Push`-1-*Storage*) is the best known moving-blocks problem. It is (one of) the hardest planning problems yet to be solved by computers, where humans still have superior performance. It has been used for decades as a testbed in artificial intelligence and has attracted a community of developers around the world that creates solvers for it. Figure 1(a) shows an instance of *Sokoban* with a movable block at $3E$, the man at $2B$, an immovable block at $1A$ and a goal location at $3B$.



(a) A *Sokoban* instance. (b) Structural information. (c) Subproblem decomposition.

Figure 1. *Sokoban* instance 1(a) and proposed methods 1(b) and 1(c).

Moving-blocks problems have the fundamental characteristics of transportation problems. Thus, if we better understand moving-blocks problems we can better understand transportation problems. The majority of the literature studied the *computational complexity* of problems with `Push` moves. Problem with `Pull` moves have been proven NP-hard while several problems with `Push` moves are PSPACE-complete. There are even fewer results for problems with `PushPull` moves. Investigating problems with `PushPull` moves may be one way to positively answer the question whether there exists an interesting but tractable version of a moving-blocks problem – in problems with `PushPull` moves every action is reversible.

According to many measures, moving-blocks problems are the hardest planning problems. They are also simple to describe and easy to implement. Thus, these problems can be used to propose new methods that are transferable to planning problems. *Heuristic search* algorithms are the most effective methods to *optimally* solve planning problems. These algorithms use an *admissible heuristic function* to guide the search. Among the most effective heuristic functions are those based on *pattern databases* (PDB heuristics). However, PDB heuristics are ineffective in transportation problems where the mapping of movable objects to goal locations is not fixed which is the case in moving-blocks problems. Also, it is common that during the search the heuristic function is not informed enough to guide the search. In general only few instances of moving-blocks problems can be optimally solved. For example, of the 90 standard instances of *Sokoban* only 10 had been optimally solved before this thesis.

2. Contributions of this Thesis

This thesis presents four major contributions listed in the subsections below which improve our understanding of the hardness and the capabilities to optimally solve of moving-blocks problems.

2.1. `Push` and `PushPull` Problems are PSPACE-complete

We investigate the computational complexity of moving-blocks problems with `Pull` and `PushPull` moves. We prove that many `Pull` problems and the entire set of `PushPull` problems is PSPACE-complete. Our reductions are from the Nondeterministic Constraint Logic (NCL) [Hearn and Demaine 2005]. We show how to build *OR* and *AND* gadgets, and how to connect them into an arbitrary planar constraint graph. Our main contribution in this research line is to enhance the knowledge of the complexity landscape of moving-blocks problems. Next, we present the main theorems in the complexity contributions. Table 1 presents a summary of our results.

Table 1. Hardness of moving-blocks problems with results of this thesis shown in boldface. Problems hard for PSPACE are also complete for PSPACE since all problems are in PSPACE.

Problem	*Storage*		*Path*	
	Hard for	Reference	Hard for	Reference
Push-1	PSPACE	[Culberson 1999]	NP	[Demaine and Hoffmann 2001]
Push-k with $k \geq 2$	Open		PSPACE	[Demaine et al. 2002]
Push-$*$	Open		PSPACE	[Demaine et al. 2002]
PushPush-1	NP	[O'Rourke 1999]	PSPACE	[Demaine et al. 2004]
PushPush-k	Open		PSPACE	[Demaine et al. 2004]
PushPush-$*$	Open		NP	[Demaine et al. 2004]
Pull-1	**PSPACE**	**[Pereira et al. 2016b]**	NP	[Ritt 2010]
Pull-k	**PSPACE**	**[Pereira et al. 2016b]**	NP	[Ritt 2010]
Pull-$*$	**PSPACE**	**[Pereira et al. 2016b]**	NP	[Ritt 2010]
PullPull-1	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PullPull-k	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PullPull-$*$	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PushPull-1	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PushPull-k	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PushPull-$*$	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PushPushPullPull-1	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PushPushPullPull-k	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**
PushPushPullPull-$*$	**PSPACE**	**[Pereira et al. 2016b]**	**PSPACE**	**[Pereira et al. 2016b]**

Theorem 1. Pull-$\{1, k, *\}$-*Storage* are PSPACE-complete.

In Pull-*Storage* problems we build the gadgets such that each movable block can be pulled to a unique goal location, prohibited moves that violate NCL constraints lead to unrecoverable states where a movable block cannot reach its goal location. Using these gadgets we show that all Pull-*Storage* problems are PSPACE-complete. Thus, now we know that these problems are at least as hard as their corresponding Push versions.

Theorem 2. PullPull-$\{1, k, *\}$-*Storage* and *Path* are PSPACE-complete.

In PullPull-*Path* problems blocks do not have goal locations. Thus, we build the gadgets such that prohibited moves leave the man trapped unable to reach its goal location in a unrecoverable state. We were able to show that PullPull-$*$ is PSPACE-complete, where the man can move an unlimited number of blocks at once while the equivalent Push version only is known to be NP-hard.

Theorem 3. PushPull-$\{1, k, *\}$-*Storage* and *Path* are PSPACE-complete.

PushPull problems do not have unrecoverable states since every move is reversible. This makes it harder to build the gadgets. Because of this, there is no distinction in our gadgets between *Storage* and *Path* versions. Considering these difficulties, surprisingly we were able to show that PushPull-1-*Path* is PSPACE-complete while the much studied Push-1-*Path* and Pull-1-*Path* are NP-hard. The gadgets used to prove the PSPACE-hardness of PushPull-1-*Path* are more complex than those used in more relaxed versions that allow to move more blocks at once. This adds to the evidence that proving PSPACE-completeness of variants where only one block can be moved is harder than proving the same hardness results for k and $*$ *Path* variants. Our proofs introduce novel ideas to NCL like buffer blocks which might help to close the knowledge gap about moving-blocks problems.

Theorem 4. `PushPushPullPull`-$\{1, k, *\}$-*Storage* and *Path* are PSPACE-complete.

We were also able to show that enforcing more constraints does not alter the complexity of these problems.

2.2. Tie-Breaking Strategies for Moving-Blocks Problems

During the search, the guidance provided by the heuristic function might not be enough to optimally solve the problem. This is especially true for hard problems like moving-blocks problems. The standard tie-breaking approach used in heuristic search prioritizes lower heuristic function values.

We propose a new tie-breaking strategy that is based on structural information of the problem. An example is shown in Figure 1(b) where our technique estimates the order in which the goal locations must be filled. Our approach is effective in many problems, for example when applied to *Pukoban* (`PushPull`-1-*Storage*). Compared to the best previous proposed tie-breaking strategy our approach more than doubles the number of optimally solved instances.

2.3. PDB Heuristics for Moving-Blocks Problems

As explained above standard PDB heuristics are ineffective in moving-blocks problems. We solve this issue by introducing an approach that decomposes the original problem into two subproblems. Then, we use the PDB heuristic to compute the cost of solving the first subproblem (white locations in Figure 1(c)) and a minimum cost perfect matching in the complete bipartite graph to compute the cost of the second subproblem.

First, we prove that when the two subproblems are independent the resulting heuristic function is admissible. Second, we prove that the heuristic function is also admissible when the two subproblems are dependent if computed according to:

$$\min_{a \in A} \left[\sum_{P \in \mathcal{B}} \delta_{a(P)}(p(P)) + \sum_{(b,g) \in M^*} \delta(a(b), g) \right] + 2L,$$

where $\delta_{a(P)}(p(P))$ is the optimal cost stored in the PDB to solve a subset of movable blocks in the first subproblem, $\delta(a(b), g)$ is the relaxed cost to solve the second subproblem where all blocks are independent and $2L$ are pairs of adjacent blocks in the optimal path of each other. To guarantee admissibility of the heuristic function we find an assignment a that minimizes the sum of the cost to solve the two subproblems.

Using the proposed heuristic function we optimally solve more instances using less the time and memory than previous methods. Considering the set of 90 standard instances, in *Pukoban* we increase the number of optimally solved instances from 20 to 45 and in *Sokoban* from 10 to 28 compared to the best previous results. Given the exponential growth in the effort to solve these instances, the increase in the number of optimally solved instances represents a major advance in our understanding of how to optimally solve extremely hard problems.

2.4. Generalization to Other Problems

We present methods to show that our contributions can be effective in other problems. First, we applied a simple version of our tie-breaking strategy to 1050 instances divided among 33 problems. Many of these problems model real-world applications. This simple version of our tie-breaking strategy increases the number of solved instances, including problems that are not transportation problems. These results indicate that stronger versions of our strategies could be used to improve the results for planning problems. Second, we also present experiments with simple versions of our heuristic functions in the *Airport* problem that models ground traffic control of airplanes in an airport. Our techniques increase the information provided by the heuristic function in the initial state, during the search and solve more instances optimally.

3. Final Remarks

Papers: We published the results of this thesis in the most prestigious conferences and journals in the areas of Artificial Intelligence and Theoretical Computer Science. In addition to the list below, two other papers with results of this thesis are being prepared for submission. The list of currently published papers follows:

- **[Pereira et al. 2016b] Theoretical Computer Science *Journal* (Qualis A1):** It is the top ranked journal on the field according to Google Scholar. The paper proves computational complexity results about moving-blocks problems.
- **[Pereira et al. 2015] Artificial Intelligence *Journal* (Qualis A1):** Considered the most important journal on Artificial Intelligence. The paper presents experimental and theoretical results about PDB heuristics and tie-breaking strategies with a focus on *Sokoban*.
- **[Pereira et al. 2016a] IJCAI *Conference* (Qualis A1):** Considered the most important conference on Artificial Intelligence. The paper introduces improved versions of our previous proposed methods.
- **[Corrêa et al. 2016] BRACIS *Conference*:** Evaluates the proposed methods on the *Airport* problem.
- **[Pereira et al. 2014b, Pereira et al. 2014a] ENIAC *Conference* and IJCAI *Doctoral Consortium*:** Introduces a PDB heuristic to detect unsolvability in *Sokoban*.
- **[Pereira et al. 2013] SoCS *Conference*:** A symposium specializing on combinatorial search. The paper presents the PDB heuristic with focus on *Sokoban*.

Impact: The impact of this thesis can be measured by the theoretical and experimental contributions to understand and optimally solve moving-blocks problems as well by its practical importance for applications. We prove computational complexity results for moving-blocks problems, including unexpected results. Our results answer open research questions and close the complexity research for some versions of the problems. Also, they show that many relaxations will not make these problems easier. Regarding optimal solutions, our methods are general and address an important limitation of PDB heuristics in transportation problems. We have currently the best published results on optimally solving *Sokoban*. This research was developed in collaboration with the well-known heuristic search group at the University of Alberta. A postdoc scholarship was granted by the Swiss Government Excellence Scholarships program to work at the University of Basel in their worldwide known planning group. However, the scholarship was declined to assume a position at UFRGS, but with the possibility of maintaining a research interaction with that group. Recent papers have been exploring techniques and research directions proposed by this thesis. For example, the paper [Asai and Fukunaga 2017] explores tie-breaking strategies for planning problems.

References

Asai, M. and Fukunaga, A. (2017). Tie-Breaking Strategies for Cost-Optimal Best First Search. *Journal of Artificial Intelligence Research*, 58:67–121.

Corrêa, A. B., Pereira, A. G., and Ritt, M. (2016). Improved airport ground traffic control with domain-dependent heuristics. In *Brazilian Conference on Intelligent Systems*, pages 73–78.

Culberson, J. (1999). Sokoban is PSPACE-complete. In *International Conference on Fun with Algorithms*, pages 65–76.

Demaine, E. D., Hearn, R. A., and Hoffmann, M. (2002). Push-2-F is PSPACE-Complete. In *Canadian Conference on Computational Geometry*, pages 31–35.

Demaine, E. D. and Hoffmann, M. (2001). Pushing blocks is NP-complete for noncrossing solution paths. In *Canadian Conference on Computational Geometry*.

Demaine, E. D., Hoffmann, M., and Holzer, M. (2004). PushPush-k is PSPACE-Complete. In *International Conference on FUN with Algorithms*, pages 159–170.

Hearn, R. and Demaine, E. D. (2005). PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96.

Hoffmann, J., Edelkamp, S., Thiébaux, S., Englert, R., dos Santos Liporace, F., and Trug, S. (2006). Engineering Benchmarks for Planning: The Domains Used in the Deterministic Part of IPC-4. *Journal of Artificial Intelligence Research*, 26(1):453–541.

Long, D. and Fox, M. (2003). The 3rd International Planning Competition: Results and Analysis. *Journal of Artificial Intelligence Research*, 20:1–59.

McDermott, D. M. (2000). The 1998 AI planning systems competition. *AI Magazine*, 21(2).

O'Rourke, J. (1999). PushPush is NP-hard in 3D. Technical report, Smith College.

Pereira, A. G., Holte, R., Schaeffer, J., Buriol, L. S., and Ritt, M. (2016a). Improved Heuristic and Tie-Breaking for Optimally Solving Sokoban. In *International Joint Conference on Artificial Intelligence*.

Pereira, A. G., Ritt, M., and Buriol, L. S. (2013). Finding Optimal Solutions to Sokoban Using Instance Dependent Pattern Databases. In *Symposium on Combinatorial Search*.

Pereira, A. G., Ritt, M., and Buriol, L. S. (2014a). Solving motion planning problems. In *International Joint Conference on Artificial Intelligence School - DC*.

Pereira, A. G., Ritt, M., and Buriol, L. S. (2014b). Solving Sokoban Optimally using Pattern Databases for Deadlock Detection. In *Encontro Nacional de Inteligência Artificial*.

Pereira, A. G., Ritt, M., and Buriol, L. S. (2015). Optimal Sokoban Solving using Pattern Databases with Specific Domain Knowledge. *Artificial Intelligence*, 227:52 – 70.

Pereira, A. G., Ritt, M., and Buriol, L. S. (2016b). Pull and PushPull are PSPACE-complete. *Theoretical Computer Science*, 628:50 – 61.

Ritt, M. (2010). Motion planning with pull moves. *CoRR*, abs/1008.2952:1–9.