Strategy Repair in Reachability Games (short paper)*

Pierre Gaillard[1,*,†], Fabio Patrizi[2,†] and Giuseppe Perelli[2,†]

[1] *ENS Paris-Saclay, University Paris-Saclay*
[2] *Sapienza University of Rome*

Abstract
We introduce *Strategy Repair*, the problem of finding a minimal amount of modifications to turn a strategy for a reachability game from losing into winning. The problem is relevant for a number of settings in Planning and Synthesis, where solutions essentially correspond to winning strategies in a suitably defined reachability game. We show, via reduction from Vertex Cover, that Strategy Repair is NP-complete and devise two algorithms, one optimal and exponential and one polynomial but sub-optimal.

Keywords
Reachability Games, Synthesis, Strategic Reasoning

1. Introduction

Reachability Games (RGs) [2] can serve as semantic models for reasoning about dynamic domains, with the resulting strategy representing the behavior that an agent can execute, in order to achieve a desired state. Typically, however, at execution time, models deviate from the actual trajectory that stems from strategy execution, resulting in a situation where the actual state does not match that of the model. There may also be situations where the goal changes during strategy execution. In both these examples, the agent is unable to keep executing the computed strategy (which was originally winning) and take appropriate actions to achieve the desired goal. Thus, the problem arises of coming up with a new strategy that guarantees goal achievement.

The original strategy might have been designed to guarantee not only goal achievement, but also a number of additional properties, such as cost minimization, reward maximization, or forbidden-state avoidance, which might yield a significant additional computational effort. Thus, when the unexpected changes are small and yield only a slightly different problem wrt the original one, i.e., only few target states are added or removed and state mismatches occur rarely, it is reasonable to seek for a solution obtained as a slight modification of the original one, under the assumption that the new strategy will retain all (or part of) the properties featured by the initial strategy, without needing the computational overhead required to achieve such properties.

This paper investigates this approach from the general perspective of RGs. We introduce a problem, called *Strategy Repair*, which requires, given a *losing* strategy σ_0, to find a minimal

ICTCS'24: Italian Conference on Theoretical Computer Science, September 11–13, 2024, Torino, Italy
* The full version appears in the proceedings of ECAI-23 [1]
*Corresponding author.
† These authors contributed equally.

amount of modifications which turn σ_0 into a winning strategy.

We make the following contributions. Firstly, we formally define the problem by introducing a notion of *distance* between two strategies, which intuitively corresponds to the number of states over which the strategies differ. Then, based on this notion, we devise a solution algorithm and characterize its complexity. Specifically, we prove, by reduction from Vertex Cover, that the decision version of Strategy Repair is NP-complete. We then investigate more efficient, but sub-optimal, alternatives, devising a polynomial greedy algorithm. The full version of this paper [1] also reports on an experimental analysis, which shows that the polynomial algorithm yields impressive results in terms of running time, scalability and accuracy (measured as distance from the optimal solution).

2. Preliminaries

A *2-player arena*, or simply *arena* is a tuple $\mathcal{A} = \langle V, V_0, V_1, \mathrm{E} \rangle$, where V is the set of *nodes*, or *vertices*, with $V = V_0 \cup V_1$ and $V_0 \cap V_1 = \emptyset$, and $\mathrm{E} \subseteq V \times V$ is the set of *edges* of the arena. We say that V_0 is the set of nodes controlled by player 0, (P_0), whereas V_1 is the set of nodes controlled by player 1 (P_1).

Definition 1 (Reachability game). *A* Reachability game *is a pair $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{A}$ is an arena, and $\mathcal{T} \subseteq V$ is a subset of nodes, sometimes called* target.

A *path* in the arena is a sequence $\pi = v_0 \cdot v_1 \cdot v_2 \ldots \in V^\omega$ such that $(v_i, v_{i+1}) \in \mathrm{E}$ for each $i \in \mathbb{N}$. As usual, by π_i, we denote the i-th node occurring in the sequence π, whereas by $\pi_{\leq i}$ we denote the prefix of π up to node π_i, also called *partial path*. We say that a path π is *winning* for player 0 if $\pi_i \in \mathcal{T}$ for some $i \in \mathbb{N}$, otherwise it is winning for player 1. A strategy for player 0 is a function $\sigma_0 : V^* \cdot V_0 \to \mathrm{E}$ mapping partial paths to edges, such that $\sigma_0(v_0 \ldots v_n)$ is an edge outgoing from v_n, for each partial path in $V^* \cdot V_0$. A strategy σ_1 for player 1 is defined accordingly. A path π is *compatible* with strategy σ_0 if $\sigma_0(\pi_{\leq i}) = (\pi_i, \pi_{i+1})$ for each $\pi_i \in V_0$. Analogously, it is *compatible* with strategy σ_1 if $\sigma_1(\pi_{\leq i}) = (\pi_i, \pi_{i+1})$ for each $\pi_i \in V_1$.

We say that a strategy σ_0 is *winning* for player 0 from v, if every path π starting from v and compatible with σ_0 is *winning*. We say that a node v is winning for player 0 if there exists a strategy σ_0 winning from v. We denote by $\mathrm{Win}_0(\mathcal{G})$ and $\mathrm{Win}_1(\mathcal{G})$ the sets of nodes in $\mathcal{G}$ that are winning for player 0 and 1, respectively. Finally, a strategy is said to be simply *winning* if it is winning from every vertex in $\mathrm{Win}_0(\mathcal{G})$. It is well known that reachability games are *memoryless determined* [3], that is, every node v is either winning for player 0 or winning for player 1 and that there always exists a memoryless winning strategy, *i.e.*, a winning strategy that is defined as $\sigma_0 : V_0 \to \mathrm{E}$ mapping each node belonging to an agent to an outgoing edge. Therefore, from now on we restrict our attention to only memoryless strategies. Such restriction allows us to define a very natural distance between two player 0 strategies σ_0 and σ_0' over the same game, that is $\mathsf{dist}(\sigma_0, \sigma_0') = |\{v \in V_0 \mid \sigma_0(v) \neq \sigma_0'(v)\}|$ Intuitively, we count the number of nodes on which the two strategies map to a different outgoing edge. This can be proved to be an actual distance [1].

We conclude this section by introducing some useful notation. For a given game $\mathcal{G}$ and an edge $e = (v_1, v_2) \in \mathrm{E}$, by $\mathcal{G}_e$ we denote the game induced from $\mathcal{G}$ by removing every edge

(v'_1, v'_2) incompatible with e, that is, such that $v'_1 = v_1$ and $v'_2 \neq v_2$. This can be extended to subsets $\mathrm{E}' \subseteq \mathrm{E}$ of edges, where $\mathcal{G}_{\mathrm{E}'} = (\mathcal{G}_{\mathrm{E}' \setminus \{e\}})_e$ is recursively defined by projecting the edges e of E' one by one. Notice that a (memoryless) strategy σ_0 can be regarded as a subset of edges, one for each node in V_0, therefore $\mathcal{G}_{\sigma_0}$ denotes the game induced from $\mathcal{G}$ by removing every edge (v, v') incompatible with σ_0, that is, such that $v \in V_0$ and $(v, v') \neq \sigma_0(v)$. Note that every vertex of V_0 has only one successor in $\mathcal{G}_{\sigma_0}$, which means that player 0 has only strategy σ_0 available in the game.

3. The Strategy Repair Problem

We now introduce the *strategy repair* problem for reachability games. First, for a given reachability game $\mathcal{G}$ and a player 0 strategy σ_0, define $\mathrm{Win}_0(\mathcal{G}, \sigma_0)$ to be the set of nodes from which σ_0 is winning. It is not hard to show that $\mathrm{Win}_0(\mathcal{G}, \sigma_0) = \mathrm{Win}_0(\mathcal{G}_{\sigma_0})$, that is, the nodes that are winning for player 0 when it is using strategy σ_0 can be obtained by considering the game $\mathcal{G}_{\sigma_0}$ where the choices incompatible with σ_0 have already been ruled out. Observe that it always holds that $\mathrm{Win}_0(\mathcal{G}, \sigma_0) \subseteq \mathrm{Win}_0(\mathcal{G})$, with $\mathrm{Win}_0(\mathcal{G}, \sigma_0) = \mathrm{Win}_0(\mathcal{G})$ if, and only if, σ_0 is winning for player 0. We define the strategy repair problem as follows.

Definition 2 (Strategy repair problem). *For a given reachability game $\mathcal{G}$ and a strategy σ_0, find a winning strategy σ'_0 such that $\mathsf{dist}(\sigma_0, \sigma'_0) \leq \mathsf{dist}(\sigma_0, \sigma''_0)$ for each winning strategy σ''_0.*

The problem introduced requires to minimize the number of modifications that are required to turn a strategy σ_0 into a strategy σ'_0 winning for a given reachability game $\mathcal{G}$. The corresponding decision problem, instead, consists in fixing a given threshold $k \in \mathbb{N}$ and checking whether some winning strategy σ'_0 exists with $\mathsf{dist}(\sigma_0, \sigma'_0) \leq k$. We now prove that the decision version of the strategy repair problem for reachability games is NP-complete. To do so, we show a reduction from the NP-complete problem *vertex cover* [4]. Given a vertex cover instance, the idea is to construct a RG with one cycle for each edge in such a way that selecting a vertex v onto the cover corresponds to one change in the strategy that breaks all the cycles corresponding to adjacent edges of v.

Theorem 1. *The strategy repair problem for reachability games is NP-complete [1].*

4. Algorithmic Solutions

We now present two algorithms for Strategy Repair, which we called `Opt` and `Greedy`, respectively. The former returns the optimal solution to the problem, but runs in exponential time. The latter, instead, returns a sub-optimal solution but runs in polynomial time. It is important to remark that they both produce correct winning strategies for the game. However, the algorithm `Greedy` does not provide the best one in terms of distance from the originally specified strategy.

We now proceed with the description of Algorithm `Opt`. In order to do so, we first introduce some useful definition. For a given game $\mathcal{G}$ and a set $X \subseteq V$ of nodes, the *Frontier* of X, denoted $\mathrm{Frontier}_0(X) = ((V_0 \setminus X) \times X) \cap \mathrm{E}$, is the set of edges that are outgoing from a Player 0 node and incoming to a node in X. Intuitively, the edges in $\mathrm{Frontier}_0$ can be used by Player 0 to enter in a single step the region X. Consider a game $\mathcal{G}$ and a strategy σ_0, and let $X = \mathrm{Win}_0(\mathcal{G}, \sigma_0)$ be the set of nodes that are winning for strategy σ_0. Observe that for an edge $(v, v') \in \mathrm{Frontier}_0(X)$, it holds that $\sigma_0(v) \neq (v, v')$, otherwise v would have been winning for σ_0 in the first place. Moreover, it is trivial to show that the strategy $\sigma'_0 = \sigma_0[v \mapsto (v, v')]$ is such that $\mathrm{Win}_0(\mathcal{G}, \sigma_0) \subsetneq \mathrm{Win}_0(\mathcal{G}, \sigma'_0)$, with the inclusion being proper because $v \in \mathrm{Win}_0(\mathcal{G}, \sigma'_0) \setminus \mathrm{Win}_0(\mathcal{G}, \sigma_0)$.

We are now ready to present the algorithm `Opt`, which is reported in Algorithm 1. The algorithm works as follow. First, it computes the winning region following σ_0 denoted $\mathrm{Win}_0(\mathcal{G}, \sigma_0)$, and compares it with the winning region of the game $\mathrm{Win}_0(\mathcal{G})$. If the two sets are equal, it means that σ_0 is already winning, so it returns the optimal solution $(\sigma_0, 0)$, with the second component denoting the cost of fixing. If that is not the case, the algorithm proceeds by first computing the frontier of $\mathrm{Win}_0(\mathcal{G}, \sigma_0)$, in order to select an edge (v, v') from it, then it compares two possible solutions. The first is obtained by solving the problem where the initial strategy is $\sigma_0[v \mapsto (v, v')]$, obtained from σ_0 by diverting the choice on v with the frontier edge (v, v'). The second is obtained by solving the problem when Player 0 is forced to select edge $\sigma_0(v)$ in v. This is obtained by considering the game $\mathcal{G}' = \mathcal{G}_{\sigma_0(v)}$, where all other outgoing edges from v are removed. Both solutions are computed with their relative costs β' and β'', which are then compared to select the best between the two. Note that the latter solution might not exist, as the choice of σ_0 in v might lead, for instance, out of the winning region. The algorithm then first checks whether such solution is viable before making a useless recursive call on $(\mathcal{G}', \sigma_0)$. Observe that in the first case the total modification cost β' must be increased by 1, as the initial strategy $\sigma_0[v \mapsto (v, v')]$ is at distance 1 from σ_0 itself. We have the following.

Algorithm 1: `Opt`.

```
Input: G a reachability game, σ0 a strategy
       for player 0
Output: Winning strategy for G
        minimizing the distance from σ0
Fix(G, σ0) :
T' ← Win0(G, σ0)
if T' = Win0(G) then
|  return (σ0, 0)
else
|  select (v, v') from Frontier(T')
|  (σ'0, β') ← Fix(G, σ0[v ↦ (v, v')])
|  G' ← G_σ0(v)
|  if v ∈ Win0(G') then
|  |  (σ''0, β'') ← Fix(G', σ0)
|  |  if β'' < β' + 1 then
|  |  |  return (σ''0, β'')
|  |  end
|  end
|  return (σ'0, β' + 1)
end
```

Theorem 2. *The algorithm* `Opt` *returns the optimal solution to the Strategy Repair problem.*

The algorithm `Opt` presented in the previous section is of exponential complexity, as it requires two recursive calls at each iteration to compare the distances between the initial strategy and two candidate best solutions. Also, notice that the recursive call that makes use of the selected edge in the frontier always computes a correct solution, although it might not be the optimal one. Therefore, a suboptimal but polynomially computable solution could be found by just selecting the one obtained from such call, disregarding the other.

This is how the algorithm `Greedy` is conceived. However, in order to improve the quality of the solution, i.e., the accuracy w.r.t. the optimum, we employ a selection criterion for the edge in the frontier set. Indeed, consider an instance $(\mathcal{G}, \sigma_0)$ of Strategy Repair, and an edge $(v, v') \in \text{Frontier}_0(\text{Win}_0(\mathcal{G}, \sigma_0))$. First, note that $\sigma_0(v) \neq (v, v')$, otherwise, the node v would be winning for σ_0 and (v, v') would not be in the frontier. Therefore, consider the set $\mathsf{Repair}_{\sigma_0}(v, v') = \text{Win}_0(\mathcal{G}, \sigma_0[v \mapsto (v, v')]) \setminus \text{Win}_0(\mathcal{G}, \sigma_0)$, that is, the set of nodes that are indirectly repaired by using the frontier edge (v, v') in the solution. Therefore, when selecting the frontier edge, one might decide to greedily maximize the number of nodes that are indirectly repaired by such a selection. This is how the algorithm `Greedy` works, as it is presented in Algorithm 2.

Algorithm 2: `Greedy`.

Input: $\mathcal{G}$ a reachability game, σ_0 a strategy for player 0
$\mathsf{Fix}(\mathcal{G}, \sigma_0)$:
$T' \leftarrow \text{Win}_0(\mathcal{G}, \sigma_0)$
if $T' = \text{Win}_0(\mathcal{G})$ **then**
 return $(\sigma_0, 0)$
end
$F \leftarrow \text{Frontier}_0(T')$
$(v, v') \leftarrow \mathsf{argmax}\{|\mathsf{Repair}_{\sigma_0}(v, v')|; (v, v') \in F\}$
$(\sigma_0', \beta') \leftarrow \mathsf{Fix}(\mathcal{G}, \sigma_0[v \mapsto (v, v')])$
return $(\sigma_0', \beta' + 1)$

5. Future Work

This work is an initial investigation into the problem of Strategy Repair and leaves at least two interesting questions open. Firstly, while the polynomial algorithm exhibits outstanding experimental performance, no approximation guarantee was obtained. For future work, we aim at studying such a property. Secondly, it is interesting to go beyond simple reachability and apply the repair approach to other problems. In particular, one immediate extension would be to investigate applicability and effectiveness of the approach for *strong cyclic* [5] solutions. More in general, the repair approach could be applied to more complex games, such as *parity* or *Büchi* games [2, 3], which would have an immediate impact on more complex forms of planning, such as Classical or FOND Planning for temporally extended goals.

Acknowledgments

Patrizi and Perelli were supported by the PNRR MUR project PE0000013-FAIR. Perelli was also supported by the PRIN 2020 projects PINPOINT and by Sapienza University of Rome under the "*Progetti Grandi di Ateneo*" programme, grant RG123188B3F7414A (ASGARD - Autonomous and Self-Governing Agent-Based Rule Design).

References

[1] P. Gaillard, F. Patrizi, G. Perelli, Strategy Repair in Reachability Games., in: K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, R. Radulescu (Eds.), ECAI'23, volume 372 of *Frontiers in Artificial*

Intelligence and Applications, IOS Press, 2023, pp. 780–787. URL: https://doi.org/10.3233/FAIA230344. doi:10.3233/FAIA230344.

[2] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], volume 2500 of *Lecture Notes in Computer Science*, Springer, 2002. URL: https://doi.org/10.1007/3-540-36387-4. doi:10.1007/3-540-36387-4.

[3] D. Perrin, J. Pin, Infinite words - automata, semigroups, logic and games, volume 141 of *Pure and applied mathematics series*, Elsevier Morgan Kaufmann, 2004.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

[5] M. Daniele, P. Traverso, M. Y. Vardi, Strong cyclic planning revisited, in: S. Biundo, M. Fox (Eds.), Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings, volume 1809 of *Lecture Notes in Computer Science*, Springer, 1999, pp. 35–48. URL: https://doi.org/10.1007/10720246_3. doi:10.1007/10720246_3.