Strategy Repair in Reachability Games

Pierre Gaillard, Fabio Patrizi, Giuseppe Perelli

Akhoury Shauryam

akhoury@cmi.ac.in

Motivation

Strategy Repair addresses a fundamental challenge in Planning and Synthesis—transforming losing strategies into winning ones within Reachability Games. The problem's significance spans various fields, from artificial intelligence to control systems.

Motivation

Strategy Repair addresses a fundamental challenge in Planning and Synthesis—transforming losing strategies into winning ones within Reachability Games. The problem's significance spans various fields, from artificial intelligence to control systems.

A reduction to Vertex Cover can be found to show it's NP-Completeness, through which two algorithms with slight modifications are devised for our problem.

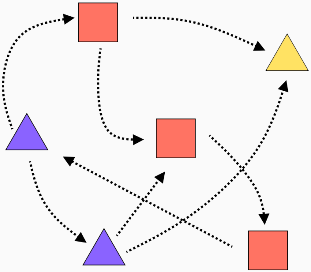
RG: Preliminaries

Introduction

A Reachability Game is a finite-state game involving two players. P_0 and P_1. Some states of the game are marked for P_0 and some for P_1. P_0 can move around in it's partitioned states until it falls on P_1's state, then the game continues with similar rules for P_1

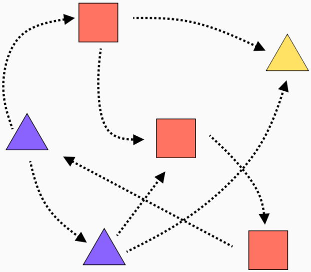
Introduction

A Reachability Game is a finite-state game involving two players. P_0 and P_1. Some states of the game are marked for P_0 and some for P_1. P_0 can move around in it's partitioned states until it falls on P_1's state, then the game continues with similar rules for P_1



Introduction

A Reachability Game is a finite-state game involving two players. P_0 and P_1. Some states of the game are marked for P_0 and some for P_1. P_0 can move around in it's partitioned states until it falls on P_1's state, then the game continues with similar rules for P_1



The Transitions from state to state are defined by the game. The goal for P_0 is to find a strategy to get to one of the Target States regardless of P_1's strategy of moving.

Definitions

An *Arena* is defined as a 4 tuple, $\mathcal{A} = \langle V, V_0, V_1, E \rangle$.

Definitions

An *Arena* is defined as a 4 tuple, $\mathcal{A} = \langle V, V_0, V_1, E \rangle$.

- V is a set of states/nodes.

Definitions

An *Arena* is defined as a 4 tuple, $\mathcal{A} = \langle V, V_0, V_1, E \rangle$.

- V is a set of states/nodes.
- V_0 and V_1 are partitions for both the players.

Definitions

An *Arena* is defined as a 4 tuple, $\mathcal{A} = \langle V, V_0, V_1, E \rangle$.

- V is a set of states/nodes.
- V_0 and V_1 are partitions for both the players.
- $E \subseteq V \times V$, the sets of Edges representing transitions.

Definitions

An *Arena* is defined as a 4 tuple, $\mathcal{A} = \langle V, V_0, V_1, E \rangle$.

- V is a set of states/nodes.
- V_0 and V_1 are partitions for both the players.
- $E \subseteq V \times V$, the sets of Edges representing transitions.

So we have $V = V_0 \cup V_1$ and $V_0 \cap V_1 = \emptyset$

Definitions

An *Arena* is defined as a 4 tuple, $\mathcal{A} = \langle V, V_0, V_1, E \rangle$.

- V is a set of states/nodes.
- V_0 and V_1 are partitions for both the players.
- $E \subseteq V \times V$, the sets of Edges representing transitions.

So we have $V = V_0 \cup V_1$ and $V_0 \cap V_1 = \emptyset$
A Reachability Game $\mathcal{G}$ is defined as $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{T} \subseteq V$ is a subset of nodes from the arena marked as the Target/Winning State.

Paths and Strategies

A path in the Arena is a sequence $\pi = \nu_0 \cdot \nu_1 \cdot \nu_2 ... \in V^\omega$. Such that each $(\nu_i, \nu_{i+1}) \in E$.

Paths and Strategies

A path in the Arena is a sequence $\pi = \nu_0 \cdot \nu_1 \cdot \nu_2 ... \in V^\omega$. Such that each $(\nu_i, \nu_{i+1}) \in E$.
Define $\pi_i = \nu_i$, and $\pi_{\leq i}$ as the partial path upto i

Paths and Strategies

A path in the Arena is a sequence $\pi = \nu_0 \cdot \nu_1 \cdot \nu_2 ... \in V^\omega$. Such that each $(\nu_i, \nu_{i+1}) \in E$.
Define $\pi_i = \nu_i$, and $\pi_{\leq i}$ as the partial path upto i

A strategy for P_0 is a function $\sigma_0 : V^* \cdot V_0 \to E$, which takes in a partial path ending in V_0 and tells which path P_0 should take.

Paths and Strategies

A path in the Arena is a sequence $\pi = \nu_0 \cdot \nu_1 \cdot \nu_2 ... \in V^\omega$. Such that each $(\nu_i, \nu_{i+1}) \in E$.
Define $\pi_i = \nu_i$, and $\pi_{\leq i}$ as the partial path upto i

A strategy for P_0 is a function $\sigma_0 : V^* \cdot V_0 \to E$, which takes in a partial path ending in V_0 and tells which path P_0 should take.

A path π and a strategy σ_0 are said to be *compatible* if $\forall \pi_i \in V_0$ we have $\sigma_0(\pi_{\leq i}) = (\pi_i, \pi_{i+1})$.

Paths and Strategies

A path in the Arena is a sequence $\pi = \nu_0 \cdot \nu_1 \cdot \nu_2 ... \in V^\omega$. Such that each $(\nu_i, \nu_{i+1}) \in E$.
Define $\pi_i = \nu_i$, and $\pi_{\leq i}$ as the partial path upto i

A strategy for P_0 is a function $\sigma_0 : V^* \cdot V_0 \to E$, which takes in a partial path ending in V_0 and tells which path P_0 should take.

A path π and a strategy σ_0 are said to be *compatible* if $\forall \pi_i \in V_0$ we have $\sigma_0(\pi_{\leq i}) = (\pi_i, \pi_{i+1})$.

Our objective is to use a 'losing' σ_0 and do minimal modifications to make it a 'winning' σ_0'. We define the metric for closeness and what is considered a 'winning' strategy further ahead.

‘OUT’ and ‘PLAY’

$\text{OUT}(\nu, \sigma_0) =$ Set of paths π that start at ν and are compatible with σ_0.

'OUT' and 'PLAY'

$\textsc{out}(\nu, \sigma_0)$ = Set of paths π that start at ν and are compatible with σ_0.

$\textsc{play}(\nu, \sigma_0, \sigma_1)$ = Set of paths starting at ν compatible with both σ_0 and σ_1. $\textsc{play}(\nu, \sigma_0, \sigma_1) = \textsc{out}(\nu, \sigma_0) \cap \textsc{out}(\nu, \sigma_1)$

‘OUT’ and ‘PLAY’

$\text{OUT}(\nu, \sigma_0) =$ Set of paths π that start at ν and are compatible with σ_0.

$\text{PLAY}(\nu, \sigma_0, \sigma_1) =$ Set of paths starting at ν compatible with both σ_0 and σ_1. $\text{PLAY}(\nu, \sigma_0, \sigma_1) = \text{OUT}(\nu, \sigma_0) \cap \text{OUT}(\nu, \sigma_1)$

σ_1 is defined similarly as σ_0 but for P_1

Winning Criterion

A path π is *winning* for P_0 if $\pi_i \in \mathcal{T}$ for some i.

Winning Criterion

A path π is *winning* for P_0 if $\pi_i \in \mathcal{T}$ for some i.

A strategy σ_0 is winning from ν if $\forall \pi \in \text{OUT}(\nu, \sigma_0)$ is winning

Winning Criterion

A path π is *winning* for P_0 if $\pi_i \in \mathcal{T}$ for some i.

A strategy σ_0 is winning from ν if $\forall \pi \in \textsc{out}(\nu, \sigma_0)$ is winning

A node ν is winning if $\exists \sigma_0$ such that σ_0 is winning from ν

Winning Criterion

A path π is *winning* for P_0 if $\pi_i \in \mathcal{T}$ for some i.

A strategy σ_0 is winning from ν if $\forall \pi \in \textsc{out}(\nu, \sigma_0)$ is winning

A node ν is winning if $\exists \sigma_0$ such that σ_0 is winning from ν

Define $\textsc{Win}_0(\mathcal{G})$ as the set of winning states for P_0 in $\mathcal{G}$

Winning Criterion

A path π is *winning* for P_0 if $\pi_i \in \mathcal{T}$ for some i.

A strategy σ_0 is winning from ν if $\forall \pi \in \textsc{out}(\nu, \sigma_0)$ is winning

A node ν is winning if $\exists \sigma_0$ such that σ_0 is winning from ν

Define $\textsc{Win}_0(\mathcal{G})$ as the set of winning states for P_0 in $\mathcal{G}$

A strategy σ_0 is winning if $\forall \nu \in \textsc{Win}_0(\mathcal{G})$, σ_0 wins from ν

The 'DIST' metric

In [D. Perrin et al 2004], the Authors showed that Reachability Games are *Memoryless Determined*. So we can get rid of the partial path leading upto V_0 in our definition for σ_0.

The ‘DIST’ metric

In [D. Perrin et al 2004], the Authors showed that Reachability Games are *Memoryless Determined*. So we can get rid of the partial path leading upto V_0 in our definition for σ_0.

Therefore, now we define $\sigma_0 : V_0 \to E$, a mapping from every Node to an Edge

The 'DIST' metric

In [D. Perrin et al 2004], the Authors showed that Reachability Games are *Memoryless Determined*. So we can get rid of the partial path leading upto V_0 in our definition for σ_0.

Therefore, now we define $\sigma_0 : V_0 \to E$, a mapping from every Node to an Edge

Now using this simpler definition, we can define a distance between two strategies σ_0 and σ_0' as

The 'DIST' metric

In [D. Perrin et al 2004], the Authors showed that Reachability Games are *Memoryless Determined*. So we can get rid of the partial path leading upto V_0 in our definition for σ_0.

Therefore, now we define $\sigma_0 : V_0 \rightarrow E$, a mapping from every Node to an Edge

Now using this simpler definition, we can define a distance between two strategies σ_0 and σ_0' as

$$\text{DIST}(\sigma_0, \sigma_0') = |\{\nu \in V | \sigma_0(\nu) \neq \sigma_0'(\nu)\}|$$

DIST is a metric

A function is a metric if it satisfies:

DIST is a metric

A function is a metric if it satisfies:

- Nonzero: $\textsc{dist}(\sigma_0, \sigma_0) = 0$

DIST is a metric

A function is a metric if it satisfies:

- Nonzero: DIST$(\sigma_0, \sigma_0) = 0$
- Positivity: DIST$(\sigma_0, \sigma_0') > 0$

DIST is a metric

A function is a metric if it satisfies:

- Nonzero: $\textsc{dist}(\sigma_0, \sigma_0) = 0$
- Positivity: $\textsc{dist}(\sigma_0, \sigma_0') > 0$
- Symmetry: $\textsc{dist}(\sigma_0, \sigma_0') = \textsc{dist}(\sigma_0', \sigma_0)$

DIST is a metric

A function is a metric if it satisfies:

- Nonzero: $\textsc{dist}(\sigma_0, \sigma_0) = 0$
- Positivity: $\textsc{dist}(\sigma_0, \sigma_0') > 0$
- Symmetry: $\textsc{dist}(\sigma_0, \sigma_0') = \textsc{dist}(\sigma_0', \sigma_0)$
- Triangle Inequality: $\textsc{dist}(\sigma_0, \sigma_0'') \leq \textsc{dist}(\sigma_0, \sigma_0') + \textsc{dist}(\sigma_0', \sigma_0'')$

DIST is a metric

A function is a metric if it satisfies:

- Nonzero: $\textsc{dist}(\sigma_0, \sigma_0) = 0$
- Positivity: $\textsc{dist}(\sigma_0, \sigma_0') > 0$
- Symmetry: $\textsc{dist}(\sigma_0, \sigma_0') = \textsc{dist}(\sigma_0', \sigma_0)$
- Triangle Inequality: $\textsc{dist}(\sigma_0, \sigma_0'') \leq \textsc{dist}(\sigma_0, \sigma_0') + \textsc{dist}(\sigma_0', \sigma_0'')$

The Triangle Inequality holds because, for every ν which doesn't match for (σ_0, σ_0'') it won't match for at least one of (σ_0, σ_0') or (σ_0', σ_0'') either. Because if it matched with both it would match for (σ_0, σ_0''). Therefore the set of such ν is contained within the other two sets.

DIST is a metric

A function is a metric if it satisfies:

- Nonzero: $\textsc{dist}(\sigma_0, \sigma_0) = 0$
- Positivity: $\textsc{dist}(\sigma_0, \sigma_0') > 0$
- Symmetry: $\textsc{dist}(\sigma_0, \sigma_0') = \textsc{dist}(\sigma_0', \sigma_0)$
- Triangle Inequality: $\textsc{dist}(\sigma_0, \sigma_0'') \leq \textsc{dist}(\sigma_0, \sigma_0') + \textsc{dist}(\sigma_0', \sigma_0'')$

The Triangle Inequality holds because, for every ν which doesn't match for (σ_0, σ_0'') it won't match for at least one of (σ_0, σ_0') or (σ_0', σ_0'') either. Because if it matched with both it would match for (σ_0, σ_0''). Therefore the set of such ν is contained within the other two sets.

It being a metric gives us a 'natural' distance between two strategies which we can use further.

Induced Game

For an edge $e = (\nu_1, \nu_2)$. Let $\mathcal{G}_e$ be the induced game we get after we remove all edges of the form (ν_1, ν_2') where $\nu_2 \neq \nu_2'$ from $\mathcal{G}$.

Induced Game

For an edge $e = (\nu_1, \nu_2)$. Let $\mathcal{G}_e$ be the induced game we get after we remove all edges of the form (ν_1, ν_2') where $\nu_2 \neq \nu_2'$ from $\mathcal{G}$.

For a subset $E' \subseteq E$, define $\mathcal{G}_{E'} = (\mathcal{G}_{E'-\{e\}})_e$ as a recursive method.

Induced Game

For an edge $e = (\nu_1, \nu_2)$. Let $\mathcal{G}_e$ be the induced game we get after we remove all edges of the form (ν_1, ν_2') where $\nu_2 \neq \nu_2'$ from $\mathcal{G}$.

For a subset $E' \subseteq E$, define $\mathcal{G}_{E'} = (\mathcal{G}_{E'-\{e\}})_e$ as a recursive method.

$\mathcal{G}_{\sigma_0}$ is the Induced Game after removing all edges not compatible with σ_0, i.e, remove (ν, ν') if $\sigma_0(\nu) \neq \nu'$

Induced Game

For an edge $e = (\nu_1, \nu_2)$. Let $\mathcal{G}_e$ be the induced game we get after we remove all edges of the form (ν_1, ν_2') where $\nu_2 \neq \nu_2'$ from $\mathcal{G}$.

For a subset $E' \subseteq E$, define $\mathcal{G}_{E'} = (\mathcal{G}_{E'-\{e\}})_e$ as a recursive method.

$\mathcal{G}_{\sigma_0}$ is the Induced Game after removing all edges not compatible with σ_0, i.e, remove (ν, ν') if $\sigma_0(\nu) \neq \nu'$

Define $\textsc{Win}(\mathcal{G}, \sigma_0)$ as the set of nodes ν from which σ_0 is winning.

Induced Game

For an edge $e = (\nu_1, \nu_2)$. Let $\mathcal{G}_e$ be the induced game we get after we remove all edges of the form (ν_1, ν_2') where $\nu_2 \neq \nu_2'$ from $\mathcal{G}$.

For a subset $E' \subseteq E$, define $\mathcal{G}_{E'} = (\mathcal{G}_{E' - \{e\}})_e$ as a recursive method.

$\mathcal{G}_{\sigma_0}$ is the Induced Game after removing all edges not compatible with σ_0, i.e, remove (ν, ν') if $\sigma_0(\nu) \neq \nu'$

Define $\textsc{Win}(\mathcal{G}, \sigma_0)$ as the set of nodes ν from which σ_0 is winning.

Now, we have:

Induced Game

For an edge $e = (\nu_1, \nu_2)$. Let $\mathcal{G}_e$ be the induced game we get after we remove all edges of the form (ν_1, ν_2') where $\nu_2 \neq \nu_2'$ from $\mathcal{G}$.

For a subset $E' \subseteq E$, define $\mathcal{G}_{E'} = (\mathcal{G}_{E'-\{e\}})_e$ as a recursive method.

$\mathcal{G}_{\sigma_0}$ is the Induced Game after removing all edges not compatible with σ_0, i.e, remove (ν, ν') if $\sigma_0(\nu) \neq \nu'$

Define $\textsc{Win}(\mathcal{G}, \sigma_0)$ as the set of nodes ν from which σ_0 is winning.

Now, we have:

- $\textsc{Win}(\mathcal{G}, \sigma_0) = \textsc{Win}(\mathcal{G}_{\sigma_0})$

Induced Game

For an edge $e = (\nu_1, \nu_2)$. Let $\mathcal{G}_e$ be the induced game we get after we remove all edges of the form (ν_1, ν_2') where $\nu_2 \neq \nu_2'$ from $\mathcal{G}$.

For a subset $E' \subseteq E$, define $\mathcal{G}_{E'} = (\mathcal{G}_{E'-\{e\}})_e$ as a recursive method.

$\mathcal{G}_{\sigma_0}$ is the Induced Game after removing all edges not compatible with σ_0, i.e, remove (ν, ν') if $\sigma_0(\nu) \neq \nu'$

Define $\textsc{Win}(\mathcal{G}, \sigma_0)$ as the set of nodes ν from which σ_0 is winning.

Now, we have:

- $\textsc{Win}(\mathcal{G}, \sigma_0) = \textsc{Win}(\mathcal{G}_{\sigma_0})$
- $Win(\mathcal{G}) = \textsc{Win}(\mathcal{G}_{\sigma_0})$ iff σ_0 is winning

Strategy Repair Problem

The Strategy Repair Problem

For a given Reachability Game $\mathcal{G}$ and a strategy σ_0. Find a winning strategy σ_0' such that $\textsc{dist}(\sigma_0, \sigma_0') \leq \textsc{dist}(\sigma_0, \sigma_0'')$ for each winning strategy σ_0''

The Strategy Repair Problem

For a given Reachability Game $\mathcal{G}$ and a strategy σ_0. Find a winning strategy σ_0' such that $\textsc{dist}(\sigma_0, \sigma_0') \leq \textsc{dist}(\sigma_0, \sigma_0'')$ for each winning strategy σ_0''

We are required to modify σ_0 with the minimum number of modifications. So the corresponding decision problem consists of fixing a threshold $k \in \mathbb{N}$ and checking if $\exists$ a winning σ_0', such that $\textsc{dist}(\sigma_0, \sigma_0') \leq k$

The Strategy Repair Problem

For a given Reachability Game $\mathcal{G}$ and a strategy σ_0. Find a winning strategy σ_0' such that $\textsc{dist}(\sigma_0, \sigma_0') \leq \textsc{dist}(\sigma_0, \sigma_0'')$ for each winning strategy σ_0''

We are required to modify σ_0 with the minimum number of modifications. So the corresponding decision problem consists of fixing a threshold $k \in \mathbb{N}$ and checking if $\exists$ a winning σ_0', such that $\textsc{dist}(\sigma_0, \sigma_0') \leq k$

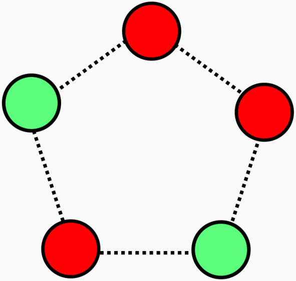
Using our definition for distance, we can find a reduction to the NP-Complete **Vertex Cover**, problem and show that the Strategy Repair Problem is NP-Complete too.

Vertex Cover Problem

For a given undirected graph $G = \langle S, A \rangle$ and a natural number $k \in \mathbb{N}$, A is a set of pair of vertices from S, find a subset $S' \subseteq S$ with $|S'| \leq k$ such that $\forall (\nu, \nu') \in A$, $\nu \in S'$ or $\nu' \in S'$

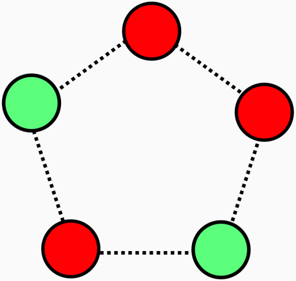
Vertex Cover Problem

For a given undirected graph $G = \langle S, A \rangle$ and a natural number $k \in \mathbb{N}$, A is a set of pair of vertices from S, find a subset $S' \subseteq S$ with $|S'| \leq k$ such that $\forall(\nu, \nu') \in A$, $\nu \in S'$ or $\nu' \in S'$



Vertex Cover Problem

For a given undirected graph $G = \langle S, A \rangle$ and a natural number $k \in \mathbb{N}$, A is a set of pair of vertices from S, find a subset $S' \subseteq S$ with $|S'| \leq k$ such that $\forall (\nu, \nu') \in A$, $\nu \in S'$ or $\nu' \in S'$



The above diagram shows a solution for a Vertx Cover problem for $k = 3$. Red nodes are in S'

SRP is in NP

For a given Reachability Game $\mathcal{G}$ and a strategy σ_0 with the threshold k, we can verify a given certificate σ_0' in Polytime.

SRP is in NP

For a given Reachability Game $\mathcal{G}$ and a strategy σ_0 with the threshold k, we can verify a given certificate σ_0' in Polytime.

Checking $\textsc{Win}(\mathcal{G}) = \textsc{Win}(G, \sigma_0')$ is linear in $|\mathcal{G}|$.

SRP is in NP

For a given Reachability Game $\mathcal{G}$ and a strategy σ_0 with the threshold k, we can verify a given certificate σ_0' in Polytime.

Checking $\textsc{Win}(\mathcal{G}) = \textsc{Win}(G, \sigma_0')$ is linear in $|\mathcal{G}|$.

And is checking whether $\textsc{dist}(\sigma_0, \sigma_0') \leq k$ or not is linear in $|V_0|$.

SRP is in NP

For a given Reachability Game $\mathcal{G}$ and a strategy σ_0 with the threshold k, we can verify a given certificate σ_0' in Polytime.

Checking $\textsc{Win}(\mathcal{G}) = \textsc{Win}(G, \sigma_0')$ is linear in $|\mathcal{G}|$.

And is checking whether $\textsc{dist}(\sigma_0, \sigma_0') \leq k$ or not is linear in $|V_0|$.

Given a certificate, we can verify it in Polytime, which concludes that Strategy Repair Problem is in NP.

Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

- $V = S \times \{0, 1\} \cup \{t\}$

Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

- $V = S \times \{0, 1\} \cup \{t\}$
- $V_0 = S \times \{0\} \cup \{t\}$

Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

- $V = S \times \{0, 1\} \cup \{t\}$
- $V_0 = S \times \{0\} \cup \{t\}$
- $V_1 = S \times \{1\}$

Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

- $V = S \times \{0, 1\} \cup \{t\}$
- $V_0 = S \times \{0\} \cup \{t\}$
- $V_1 = S \times \{1\}$
- $\mathcal{T} = \{t\}$

Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

- $V = S \times \{0, 1\} \cup \{t\}$
- $V_0 = S \times \{0\} \cup \{t\}$
- $V_1 = S \times \{1\}$
- $\mathcal{T} = \{t\}$
- $E_0 = \{((\nu, 1), (\nu, 0)) | \nu, \nu' \in A\}$

Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

- $V = S \times \{0, 1\} \cup \{t\}$
- $V_0 = S \times \{0\} \cup \{t\}$
- $V_1 = S \times \{1\}$
- $\mathcal{T} = \{t\}$
- $E_0 = \{((\nu, 1), (\nu, 0)) | \nu, \nu' \in A\}$
- $E = E_0 \cup \{((\nu, 0), (\nu, 1)) | \nu \in S\} \cup \{((\nu, 0), t) | \nu \in S\}$

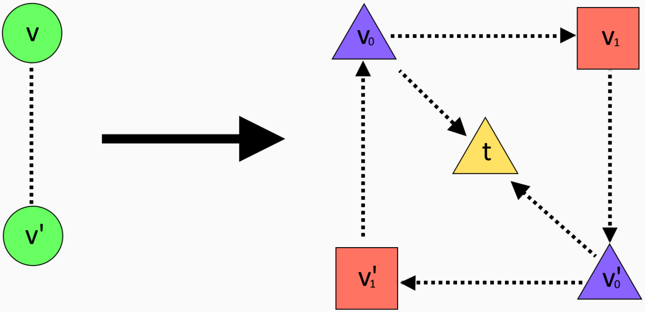
Reduction with Vertex Cover

Consider an instance of the Vertex Cover problem with $G = \langle S, A \rangle$ and threshold k given to us. Define an RG $\mathcal{G} = \langle \mathcal{A}, \mathcal{T} \rangle, \sigma_0, k$ with $\mathcal{A} = \langle V, V_0, V_1, E \rangle$ for a fresh new node t as:

- $V = S \times \{0, 1\} \cup \{t\}$
- $V_0 = S \times \{0\} \cup \{t\}$
- $V_1 = S \times \{1\}$
- $\mathcal{T} = \{t\}$
- $E_0 = \{((\nu, 1), (\nu, 0)) | \nu, \nu' \in A\}$
- $E = E_0 \cup \{((\nu, 0), (\nu, 1)) | \nu \in S\} \cup \{((\nu, 0), t) | \nu \in S\}$
- $\sigma_0((\nu, 0))$ sends to $(\nu, 1)$

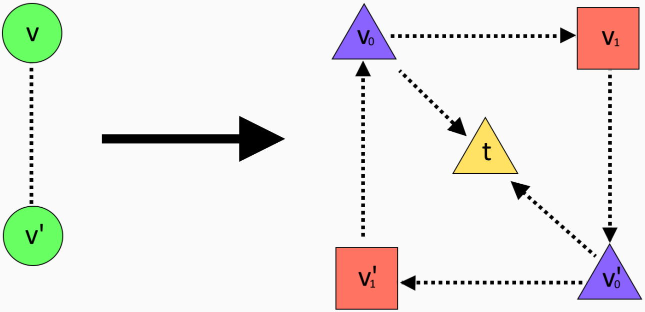
Reduction with Vertex Cover

Example of a reduction



Reduction with Vertex Cover

Example of a reduction



Each node ν corresponds to 2 new nodes in the RG, here we write them as ν_0 and ν_1

Proof

We now need to show the undirected graph G has a Vertex Cover S' of size at most k if and only if the subsequent SRP has a winning σ_0' such that $\textsc{dist}(\sigma_0, \sigma_0') \leq k$

Proof

We now need to show the undirected graph G has a Vertex Cover S' of size at most k if and only if the subsequent SRP has a winning σ'_0 such that $\textsc{dist}(\sigma_0, \sigma'_0) \leq k$

For simplicity, let $(\nu, 0)$ be dentoed as ν_0 and $(\nu, 1)$ as ν_1.

Proof

We now need to show the undirected graph G has a Vertex Cover S' of size at most k if and only if the subsequent SRP has a winning σ_0' such that $\textsc{dist}(\sigma_0, \sigma_0') \leq k$

For simplicity, let $(\nu, 0)$ be dentoed as ν_0 and $(\nu, 1)$ as ν_1.

Observe that $\textsc{Win}_0(\mathcal{G}) = V$ as the strategy that selects all edges going to the target t is clearly winning from every node of the game.

Proof-Forwards

Suppose we have a solution S' for the Vertex Cover Problem. So, $|S'| \leq k$.

Proof-Forwards

Suppose we have a solution S' for the Vertex Cover Problem. So, $|S'| \leq k$.

Consider the strategy $\sigma_0' = \sigma_0[\nu_0 \mapsto (\nu_0, t) | \nu \in S']$

Proof-Forwards

Suppose we have a solution S' for the Vertex Cover Problem. So, $|S'| \leq k$.

Consider the strategy $\sigma_0' = \sigma_0[\nu_0 \mapsto (\nu_0, t) | \nu \in S']$

We have modified $_0$ for each element in S', therefore $\textsc{dist}(\sigma_0, \sigma_0') = |S'| \leq k$

Proof-Forwards

Suppose we have a solution S' for the Vertex Cover Problem. So, $|S'| \leq k$.

Consider the strategy $\sigma_0' = \sigma_0[\nu_0 \mapsto (\nu_0, t) | \nu \in S']$

We have modified $_0$ for each element in S', therefore $\textsc{dist}(\sigma_0, \sigma_0') = |S'| \leq k$

Now we need to show that σ_0' is winning everywhere.

Proof-Forwards: Continued

It is obvious that $t \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$.

Proof-Forwards: Continued

It is obvious that $t \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$. As for the rest of the nodes, they can be distinguished into 4 categories:

Proof-Forwards: Continued

It is obvious that $t \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$. As for the rest of the nodes, they can be distinguished into 4 categories:

1. $\nu_0 \in V_0$ and $\nu \in S'$, $\sigma_0'(\nu_0) = (\nu_0, t)$, $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$

Proof-Forwards: Continued

It is obvious that $t \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$. As for the rest of the nodes, they can be distinguished into 4 categories:

1. $\nu_0 \in V_0$ and $\nu \in S'$, $\sigma_0'(\nu_0) = (\nu_0, t)$, $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
2. $\nu_1 \in V_1$ and $\nu \notin S'$, every successor of ν_1 is some ν_0' that has $\nu' \in S$. From Case 1, it follows that $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$

Proof-Forwards: Continued

It is obvious that $t \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$. As for the rest of the nodes, they can be distinguished into 4 categories:

1. $\nu_0 \in V_0$ and $\nu \in S'$, $\sigma_0'(\nu_0) = (\nu_0, t)$, $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
2. $\nu_1 \in V_1$ and $\nu \notin S'$, every successor of ν_1 is some ν_0' that has $\nu' \in S$. From Case 1, it follows that $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
3. $\nu_0 \in V_0$ and $\nu \notin S'$, $\sigma_0'(\nu_0) = (\nu_0, \nu_1)$, sends us to $\nu_1 \in V_1$. From Case 2, $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$ which implies $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$

Proof-Forwards: Continued

It is obvious that $t \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$. As for the rest of the nodes, they can be distinguished into 4 categories:

1. $\nu_0 \in V_0$ and $\nu \in S'$, $\sigma_0'(\nu_0) = (\nu_0, t)$, $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
2. $\nu_1 \in V_1$ and $\nu \notin S'$, every successor of ν_1 is some ν_0' that has $\nu' \in S$. From Case 1, it follows that $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
3. $\nu_0 \in V_0$ and $\nu \notin S'$, $\sigma_0'(\nu_0) = (\nu_0, \nu_1)$, sends us to $\nu_1 \in V_1$. From Case 2, $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$ which implies $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
4. $\nu_1 \in V_1$ and $\nu \in S'$, every successor is of form $\nu_0' \in V_0$, so through Case 1 or Case 3, $\nu_0' \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$, therefore $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$

Proof-Forwards: Continued

It is obvious that $t \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$. As for the rest of the nodes, they can be distinguished into 4 categories:

1. $\nu_0 \in V_0$ and $\nu \in S'$, $\sigma_0'(\nu_0) = (\nu_0, t)$, $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
2. $\nu_1 \in V_1$ and $\nu \notin S'$, every successor of ν_1 is some ν_0' that has $\nu' \in S$. From Case 1, it follows that $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
3. $\nu_0 \in V_0$ and $\nu \notin S'$, $\sigma_0'(\nu_0) = (\nu_0, \nu_1)$, sends us to $\nu_1 \in V_1$. From Case 2, $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$ which implies $\nu_0 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$
4. $\nu_1 \in V_1$ and $\nu \in S'$, every successor is of form $\nu_0' \in V_0$, so through Case 1 or Case 3, $\nu_0' \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$, therefore $\nu_1 \in \textsc{Win}_0(\mathcal{G}, \sigma_0')$

Hence the new strategy is winning from every state $\rightarrow \sigma_0'$ is a winning strategy.

Proof-Backwards

Now, assume we have a winning strategy σ_0' to SPR.

Proof-Backwards

Now, assume we have a winning strategy σ_0' to SPR.

This means $\textsc{dist}(\sigma_0, \sigma_0') \leq k$.

Proof-Backwards

Now, assume we have a winning strategy σ_0' to SPR.

This means $\textsc{dist}(\sigma_0, \sigma_0') \leq k$.

Define $S' = \{\nu | \sigma_0(\nu) \neq \sigma_0'(\nu)\}$.

Proof-Backwards

Now, assume we have a winning strategy σ'_0 to SPR.

This means $\textsc{dist}(\sigma_0, \sigma'_0) \leq k$.

Define $S' = \{\nu | \sigma_0(\nu) \neq \sigma'_0(\nu)\}$.

This implies that $|S'| = \textsc{dist}(\sigma_0, \sigma'_0) \leq k$.

Proof-Backwards

Now, assume we have a winning strategy σ_0' to SPR.

This means $\textsc{dist}(\sigma_0, \sigma_0') \leq k$.

Define $S' = \{\nu | \sigma_0(\nu) \neq \sigma_0'(\nu)\}$.

This implies that $|S'| = \textsc{dist}(\sigma_0, \sigma_0') \leq k$.

We need to show that S' is a valid solution to the Vertex Cover Problem.

Proof-Backwards: Continued

Assume on the contrary, that there exists an edge $\{\nu, \nu'\}$ is not covered S'.

Proof-Backwards: Continued

Assume on the contrary, that there exists an edge $\{\nu, \nu'\}$ is not covered S'.

Therefore $\sigma_0'(\nu_0) = (\nu_0, \nu_1)$ and $\sigma_0'(\nu_0') = (\nu_0', \nu_1')$

Proof-Backwards: Continued

Assume on the contrary, that there exists an edge $\{\nu, \nu'\}$ is not covered S'.

Therefore $\sigma_0'(\nu_0) = (\nu_0, \nu_1)$ and $\sigma_0'(\nu_0') = (\nu_0', \nu_1')$

If P_1 sets $\sigma_1(\nu_1) = (\nu_1, \nu_0')$ and $\sigma_1(\nu_1') = (\nu_1', \nu_0)$.

Proof-Backwards: Continued

Assume on the contrary, that there exists an edge $\{\nu, \nu'\}$ is not covered S'.

Therefore $\sigma_0'(\nu_0) = (\nu_0, \nu_1)$ and $\sigma_0'(\nu_0') = (\nu_0', \nu_1')$

If P_1 sets $\sigma_1(\nu_1) = (\nu_1, \nu_0')$ and $\sigma_1(\nu_1') = (\nu_1', \nu_0)$.

This creates a loop within $\nu_0, \nu_1, \nu_0', \nu_1'$ and we never reach a winning state.

Proof-Backwards: Continued

Assume on the contrary, that there exists an edge $\{\nu, \nu'\}$ is not covered S'.

Therefore $\sigma_0'(\nu_0) = (\nu_0, \nu_1)$ and $\sigma_0'(\nu_0') = (\nu_0', \nu_1')$

If P_1 sets $\sigma_1(\nu_1) = (\nu_1, \nu_0')$ and $\sigma_1(\nu_1') = (\nu_1', \nu_0)$.

This creates a loop within $\nu_0, \nu_1, \nu_0', \nu_1'$ and we never reach a winning state.

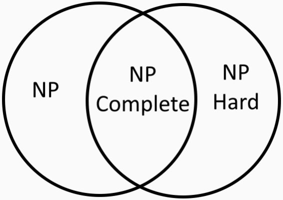
But, σ_0' was given to us as a winning strategy, this is a contradiction, therefore our assumption was wrong, i.e, S' is a valid solution to our Vertex Cover.

SRP is NP-Complete

We've shown a reduction for Strategy Repair Problem to Vertex Cover Problem, which is an NP-Complete problem, this means the Strategy Repair Problem is NP-Hard.

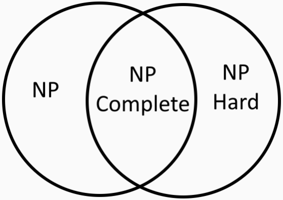
SRP is NP-Complete

We've shown a reduction for Strategy Repair Problem to Vertex Cover Problem, which is an NP-Complete problem, this means the Strategy Repair Problem is NP-Hard.



SRP is NP-Complete

We've shown a reduction for Strategy Repair Problem to Vertex Cover Problem, which is an NP-Complete problem, this means the Strategy Repair Problem is NP-Hard.



We have already seen how SRP is itself in NP, so using the two we can conculde that SRP is NP-Complete itself. Now, we can explore some algorithms for SRP

Algorithms

FRONTIER and REPAIR

Define $\textsc{Frontier}_0(X) = ((V_0 - X) \times X) \cap E$, the set of outgoing edges from V_0 to a node in X, for some $X \subseteq V$.

FRONTIER and REPAIR

Define $\textsc{Frontier}_0(X) = ((V_0 - X) \times X) \cap E$, the set of outgoing edges from V_0 to a node in X, for some $X \subseteq V$.

Let $\textsc{Repair}_{\sigma_0}(\nu, \nu') = \textsc{Win}_0(\mathcal{G}, \sigma_0[\nu \mapsto (\nu, \nu')]) - \textsc{Win}_0(\mathcal{G}, \sigma_0)$

Optimal

For a game $\mathcal{G}$ and a strategy σ_0. Let $X = \textsc{Win}_0(\mathcal{G}, \sigma_0)$.

Optimal

For a game $\mathcal{G}$ and a strategy σ_0. Let $X = \textsc{Win}_0(\mathcal{G}, \sigma_0)$.

Then for an edge $(\nu, \nu') \in \textsc{Frontier}_0(X)$, $\sigma_0(\nu) \neq (\nu, \nu')$ since otherwise it would've been winning and not be in the Frontier.

Optimal

For a game $\mathcal{G}$ and a strategy σ_0. Let $X = \textsc{Win}_0(\mathcal{G}, \sigma_0)$.

Then for an edge $(\nu, \nu') \in \textsc{Frontier}_0(X)$, $\sigma_0(\nu) \neq (\nu, \nu')$ since otherwise it would've been winning and not be in the Frontier.

The strategy $\sigma_0' = \sigma_0[\nu \mapsto (\nu, \nu')]$ has the property $\textsc{Win}_0(\mathcal{G}, \sigma_0) \subset \textsc{Win}_0(\mathcal{G}, \sigma_0')$ and $\nu \in \textsc{Win}_0(\mathcal{G}, \sigma_0') - \textsc{Win}_0(\mathcal{G}, \sigma_0)$

Optimal

For a game $\mathcal{G}$ and a strategy σ_0. Let $X = \textsc{Win}_0(\mathcal{G}, \sigma_0)$.

Then for an edge $(\nu, \nu') \in \textsc{Frontier}_0(X)$, $\sigma_0(\nu) \neq (\nu, \nu')$ since otherwise it would've been winning and not be in the Frontier.

The strategy $\sigma_0' = \sigma_0[\nu \mapsto (\nu, \nu')]$ has the property $\textsc{Win}_0(\mathcal{G}, \sigma_0) \subset \textsc{Win}_0(\mathcal{G}, \sigma_0')$ and $\nu \in \textsc{Win}_0(\mathcal{G}, \sigma_0') - \textsc{Win}_0(\mathcal{G}, \sigma_0)$

Using these facts, we can create an algorithm to optimize our strategy at each step.

Optimal: Code

```
Data: $\mathcal{G}$ a Reachability Game and a Strategy $\sigma_0$
Result: Closest Winning Strategy
FIX($\mathcal{G}, \sigma_0$)
$T' \leftarrow$ WIN$_0(\mathcal{G}, \sigma_0)$
if $T' = WIN_0(\mathcal{G})$ then
  | RETURN($\sigma_0, 0$)
else
  | select $(\nu, \nu')$ from FRONTIER($T'$)
  | $(\sigma_0', \beta') \leftarrow$ FIX($\mathcal{G}, \sigma_0[\nu \mapsto (\nu, \nu')]$)
  | $\mathcal{G}' \leftarrow \mathcal{G}_{\sigma_0(\nu)}$
  | if $\nu \in WIN_0(\mathcal{G}')$ then
  |   | $(\sigma_0'', \beta'') \leftarrow$ FIX($\mathcal{G}', \sigma_0$)
  |   | if $\beta'' < \beta + 1$ then
  |   |   | RETURN($\sigma_0'', \beta''$)
  |   | end
  | end
  | RETURN($\sigma_0', \beta' + 1$)
end
```

Greedy

The Greedy algorithm is made because Optimal is of exponential complexity as it uses two recursive calls each iteration.

Greedy

The Greedy algorithm is made because Optimal is of exponential complexity as it uses two recursive calls each iteration.

Choosing from the Frontier set is slow, so we must specify what to choose, but that might reduce accuracy, therefore we employ a better selection criterion.

Greedy: Cont

Consider an edge $(\nu, \nu') \in \textsc{Frontier}_0(\textsc{Win}_0(\mathcal{G}, \sigma_0))$. Note that $\sigma_0(\nu) \neq (\nu, \nu')$

Greedy: Cont

Consider an edge $(\nu, \nu') \in \textsc{Frontier}_0(\textsc{Win}_0(\mathcal{G}, \sigma_0))$. Note that $\sigma_0(\nu) \neq (\nu, \nu')$

$$\textsc{Repair}_{\sigma_0}(\nu, \nu') = \textsc{Win}_0(\mathcal{G}, \sigma_0[\nu \mapsto (\nu, \nu')]) - \textsc{Win}_0(\mathcal{G}, \sigma_0)$$

Greedy: Cont

Consider an edge $(\nu, \nu') \in \textsc{Frontier}_0(\textsc{Win}_0(\mathcal{G}, \sigma_0))$. Note that $\sigma_0(\nu) \neq (\nu, \nu')$

$$\textsc{Repair}_{\sigma_0}(\nu, \nu') = \textsc{Win}_0(\mathcal{G}, \sigma_0[\nu \mapsto (\nu, \nu')]) - \textsc{Win}_0(\mathcal{G}, \sigma_0)$$

Observe the set $\textsc{Repair}_{\sigma_0}(\nu, \nu')$, set of nodes that are indirectly repaired by using the frontier edge.

Greedy: Cont

Consider an edge $(\nu, \nu') \in \textsc{Frontier}_0(\textsc{Win}_0(\mathcal{G}, \sigma_0))$. Note that $\sigma_0(\nu) \neq (\nu, \nu')$

$$\textsc{Repair}_{\sigma_0}(\nu, \nu') = \textsc{Win}_0(\mathcal{G}, \sigma_0[\nu \mapsto (\nu, \nu')]) - \textsc{Win}_0(\mathcal{G}, \sigma_0)$$

Observe the set $\textsc{Repair}_{\sigma_0}(\nu, \nu')$, set of nodes that are indirectly repaired by using the frontier edge.

Choosing the edge that maximises the number of repaired nodes results in the Greedy Algorithm

Greedy: Code

Data: $\mathcal{G}$ a Reachability Game and a Strategy σ_0

Result: A Winning Strategy

$\textsc{Fix}(\mathcal{G}, \sigma_0)$

$T' \leftarrow \textsc{Win}_0(\mathcal{G}, \sigma_0)$

if $T' = \textsc{Win}_0(\mathcal{G})$ **then**

  $\textsc{return}(\sigma_0, 0)$

else

  $F \leftarrow \textsc{Frontier}_0(T')$

  $(\nu, \nu') \leftarrow \text{argmax}\{|\textsc{Repair}_{\sigma_0}(\nu, \nu')| (\nu, \nu') \in F\}$

  $(\sigma_0', \beta') \leftarrow \textsc{Fix}(\mathcal{G}, \sigma_0[\nu \mapsto (\nu, \nu')])$

  $\textsc{return}(\sigma_0', \beta' + 1)$

end

MustFix

The Greedy algorithm selects frontier edges to maximize the number of nodes entering the winning area during the repair process. However, this approach can always be applied, as any strategy needing repair implies the existence of at least one suitable edge for selection.

MustFix

The Greedy algorithm selects frontier edges to maximize the number of nodes entering the winning area during the repair process. However, this approach can always be applied, as any strategy needing repair implies the existence of at least one suitable edge for selection.

Introducing the MustFix method, it specifically targets the selection of frontier edges crucial for repairing the current strategy.

MustFix

The Greedy algorithm selects frontier edges to maximize the number of nodes entering the winning area during the repair process. However, this approach can always be applied, as any strategy needing repair implies the existence of at least one suitable edge for selection.

Introducing the MustFix method, it specifically targets the selection of frontier edges crucial for repairing the current strategy.

If a frontier edge (ν, ν') in $\textsc{Frontier}_0(\textsc{Win}_0(\mathcal{G}, \sigma_0))$ satisfies the condition $\nu \notin \textsc{Win}_0(\mathcal{G}_{\sigma_0(\nu)})$, indicating that no consistent strategy with $\sigma_0(\nu)$ is winning for ν.

MustFix

The Greedy algorithm selects frontier edges to maximize the number of nodes entering the winning area during the repair process. However, this approach can always be applied, as any strategy needing repair implies the existence of at least one suitable edge for selection.

Introducing the MustFix method, it specifically targets the selection of frontier edges crucial for repairing the current strategy.

If a frontier edge (ν, ν') in $\textsc{Frontier}_0(\textsc{Win}_0(\mathcal{G}, \sigma_0))$ satisfies the condition $\nu \notin \textsc{Win}_0(\mathcal{G}_{\sigma_0(\nu)})$, indicating that no consistent strategy with $\sigma_0(\nu)$ is winning for ν.

MustFix selects such edges for the repair process. This targeted selection improves efficiency by focusing on edges essential for strategy correction.

MustFix: Cont

Two obervations are noteworthy:

MustFix: Cont

Two obervations are noteworthy:

1. MustFix method doesn't guarantee the existence of frontier edges meeting its conditions. Therefore, other selection methods, like those in the Greedy algorithm, are still necessary.

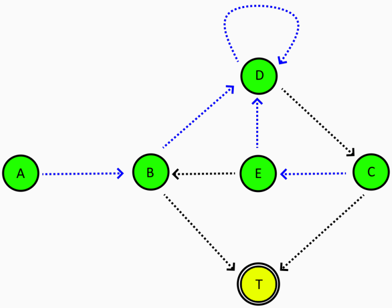
MustFix: Cont

Two obervations are noteworthy:

1. MustFix method doesn't guarantee the existence of frontier edges meeting its conditions. Therefore, other selection methods, like those in the Greedy algorithm, are still necessary.
2. MustFix can be employed as a preprocessing mechanism for Opt. It identifies edges essential even in the optimal solution, streamlining the process by requiring only one recursive call when such edges are selected.

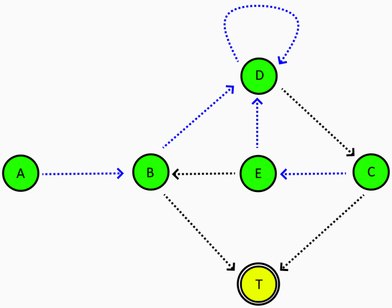
Greedy isn't Minimal

Suppose, the blue edges below denote the strategy's choices. t denotes the target.



Greedy isn't Minimal

Suppose, the blue edges below denote the strategy's choices. t denotes the target.



Deploying Greedy, even with MustFix results in 3 modifications. Whereas the closest winning strategy is 2 modifications away. $\sigma_0' = \sigma_0[C \mapsto (C, t), D \mapsto (D, C)]$

Results

Setup

The random instances were generated as follows:

Setup

The random instances were generated as follows:

- For input n, the random generator produced a game with n nodes.

Setup

The random instances were generated as follows:

- For input n, the random generator produced a game with n nodes.
- Each node was assigned to P_0 or P_1 with equal probability.

Setup

The random instances were generated as follows:

- For input n, the random generator produced a game with n nodes.
- Each node was assigned to P_0 or P_1 with equal probability.
- For each node, a number d was generated with constraints. Then, d successors were then randomly assigned to the node.

Setup

The random instances were generated as follows:

- For input n, the random generator produced a game with n nodes.
- Each node was assigned to P_0 or P_1 with equal probability.
- For each node, a number d was generated with constraints. Then, d successors were then randomly assigned to the node.

Additionally, t of target vertices were uniformly selected. t is fixed to be 5% of n.

Setup

The random instances were generated as follows:

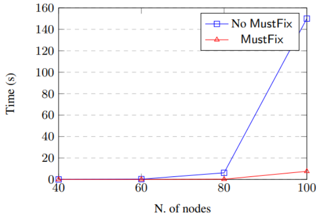
- For input n, the random generator produced a game with n nodes.
- Each node was assigned to P_0 or P_1 with equal probability.
- For each node, a number d was generated with constraints. Then, d successors were then randomly assigned to the node.

Additionally, t of target vertices were uniformly selected. t is fixed to be 5% of n.

Each Game was run on Optimal (with and without MustFix) and Greedy with MustFix.

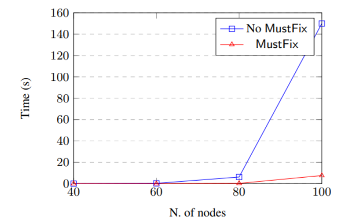
Opt Results

N. nodes	N. experiments	no MustFix	MustFix
40	1000	0.0043	0.0017
60	100	0.28	0.013
80	20	6.2	0.25
100	20	150	7.6



Opt Results

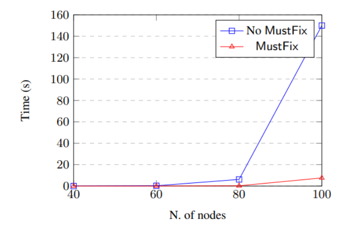
N. nodes	N. experiments	no MustFix	MustFix
40	1000	0.0043	0.0017
60	100	0.28	0.013
80	20	6.2	0.25
100	20	150	7.6



These are the results for Opt on different sized games, with and without MustFix and the time they took.

Opt Results

N. nodes	N. experiments	no MustFix	MustFix
40	1000	0.0043	0.0017
60	100	0.28	0.013
80	20	6.2	0.25
100	20	150	7.6

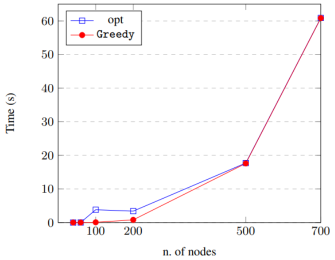


These are the results for Opt on different sized games, with and without MustFix and the time they took.

The results are to be expected since MustFix tries to avoid the recursive call whenever possible

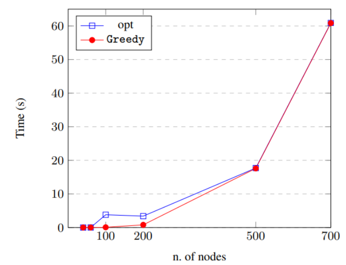
Opt vs Greedy

N. nodes	N. experiments	Opt	Greedy	Accuracy
40	5000	0.0012	0.001	0.9994
60	5000	0.02	0.0065	0.9952
100	1000	3.8	0.064	0.9904
200	700	3.39	0.79	0.9994
500	300	17.71	17.6	1
700	150	60.9	60.85	1



Opt vs Greedy

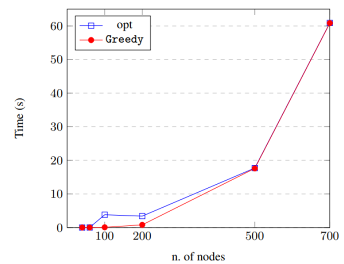
N. nodes	N. experiments	Opt	Greedy	Accuracy
40	5000	0.0012	0.001	0.9994
60	5000	0.02	0.0065	0.9952
100	1000	3.8	0.064	0.9904
200	700	3.39	0.79	0.9994
500	300	17.71	17.6	1
700	150	60.9	60.85	1



Accuracy is defined as $\frac{\mathsf{DIST}(\sigma_0, \sigma_0^{Opt})}{\mathsf{DIST}(\sigma_0, \sigma_0^{Greedy})}$.

Opt vs Greedy

N. nodes	N. experiments	Opt	Greedy	Accuracy
40	5000	0.0012	0.001	0.9994
60	5000	0.02	0.0065	0.9952
100	1000	3.8	0.064	0.9904
200	700	3.39	0.79	0.9994
500	300	17.71	17.6	1
700	150	60.9	60.85	1



Accuracy is defined as $\frac{\text{DIST}(\sigma_0,\sigma_0^{Opt})}{\text{DIST}(\sigma_0,\sigma_0^{Greedy})}$.

Since MustFix is applied to both, they both take roughly the same amount of time for very big games.

Conclusions

Summary

The authors introduced Strategy Repair, aiming to repair a losing strategy for a Reachability Game into a winning one with minimal modifications. Ensuring a minimal-distance solution with respect to strategy changes, was proven NP-complete through a reduction from Vertex Cover.

Summary

The authors introduced Strategy Repair, aiming to repair a losing strategy for a Reachability Game into a winning one with minimal modifications. Ensuring a minimal-distance solution with respect to strategy changes, was proven NP-complete through a reduction from Vertex Cover.

To handle practical high complexity, the authors devised a polynomial, greedy algorithm and an efficient heuristic named MustFix. MustFix demonstrated remarkable practical efficiency in terms of both runtime and strategy modification distance. Even on randomly generated problems, the optimal algorithm closely rivaled the suboptimal, polynomial one.

Future Work

While the polynomial algorithm, along with the MustFix heuristic, demonstrates excellent experimental performance, no approximation guarantee was achieved. Future work can focus on exploring and establishing such a guarantee.

Future Work

While the polynomial algorithm, along with the MustFix heuristic, demonstrates excellent experimental performance, no approximation guarantee was achieved. Future work can focus on exploring and establishing such a guarantee.

It holds potential for complex games like parity or Büchi games, impacting advanced planning scenarios such as Classical or Fully Observable Non-Deterministic (FOND) Planning for temporally extended goals.

Thank You!