CE810 - Game Design 2

Artificial Intelligence

Joseph Walton-Rivers & Piers Williams

Monday, 21 May 2018

University of Essex

Artificial Intelligence

- Production Rule Agents

Artificial Intelligence

- Production Rule Agents
- Monte-Carlo Tree Search

Artificial Intelligence

- Production Rule Agents
- Monte-Carlo Tree Search
- Genetic Algorithms

Artificial Intelligence

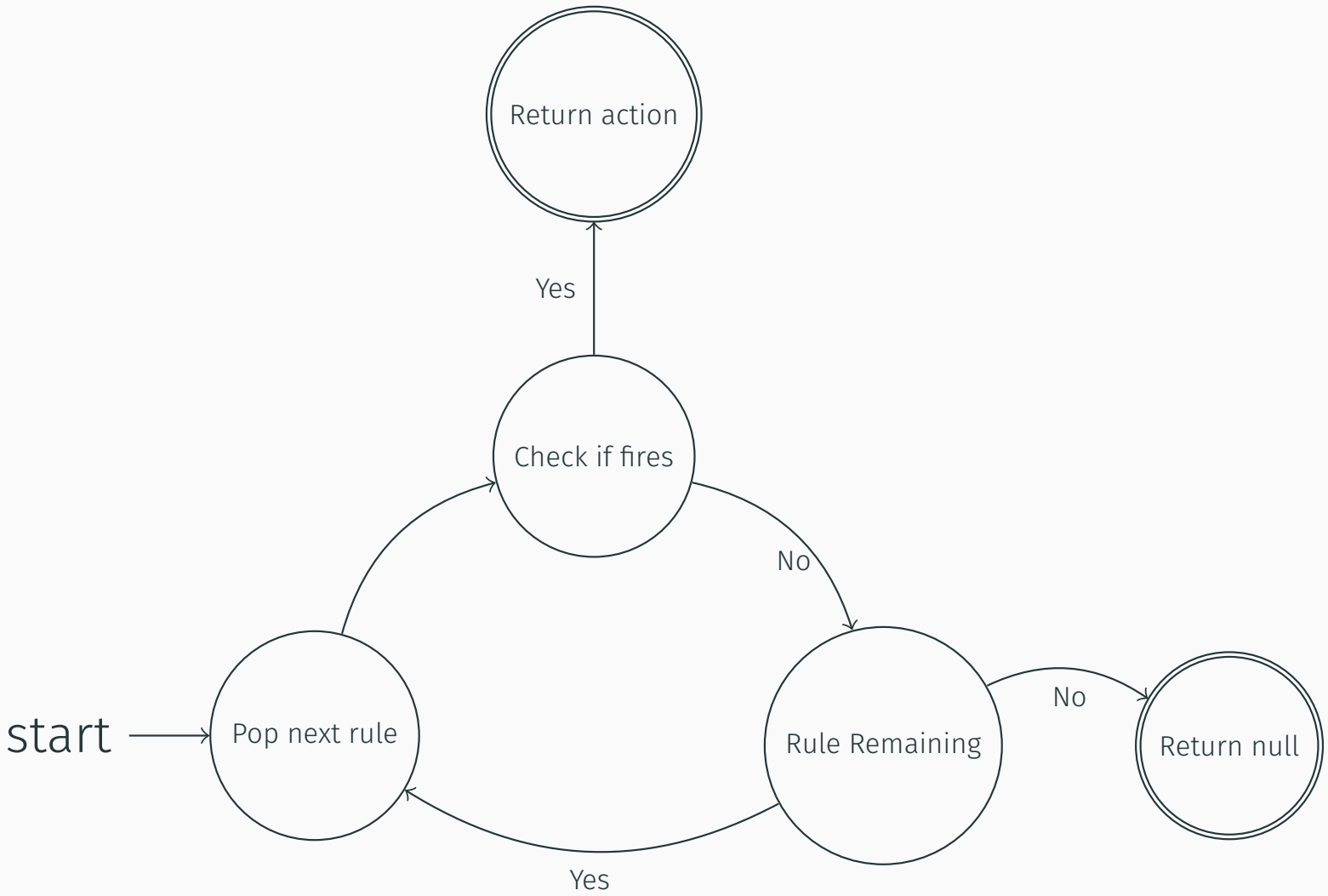
- Production Rule Agents
- Monte-Carlo Tree Search
- Genetic Algorithms
- Neural Networks

Production Rule Agents

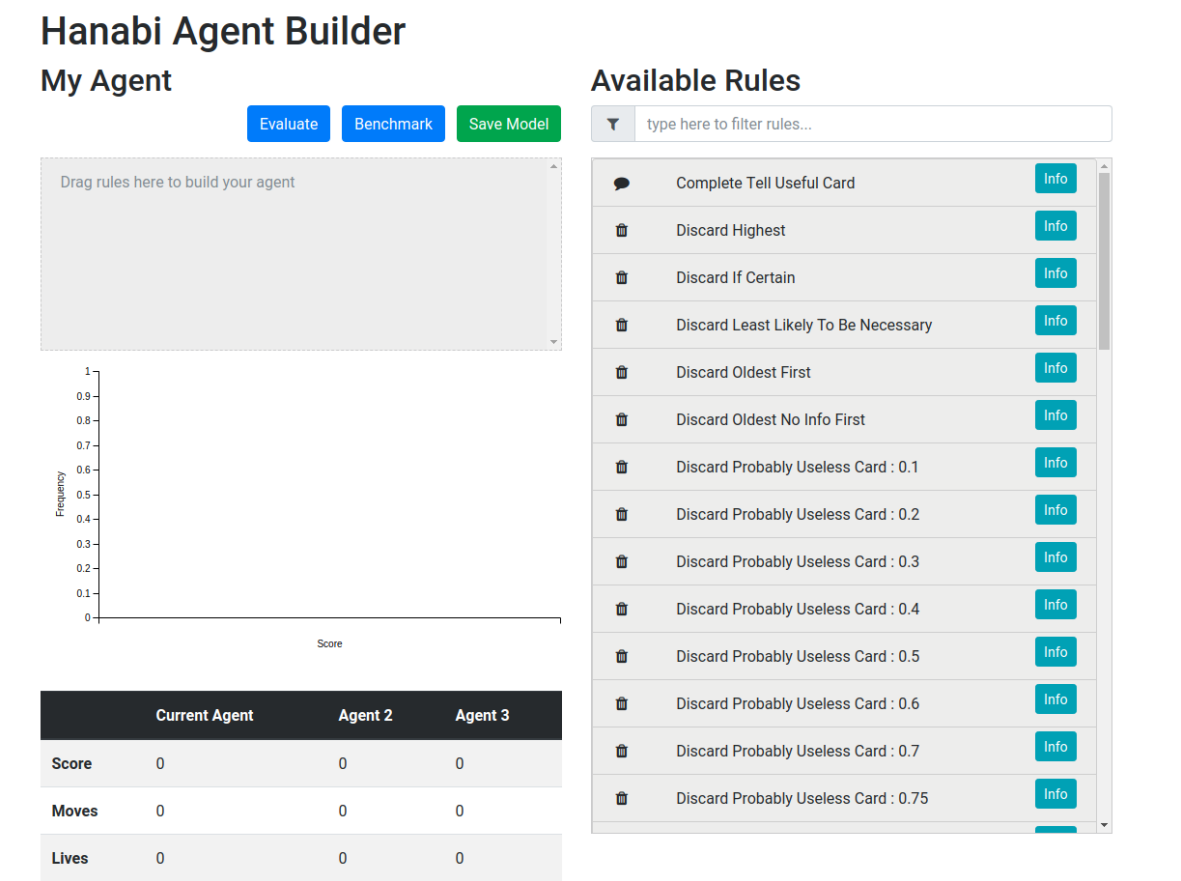
Basic

```java
@FunctionalInterface
public interface Rule {
  boolean couldFire(int playerID, GameState state);
  Action execute(int playerID, GameState state);
}
```

Diagram


Return action
Yes
Check if fires
No
start
Pop next rule
Rule Remaining
No
Return null
Yes

Hanabi



Exercise

Try it out!

Basic-TBS

```
public interface ProductionRule {

Map<UUID, Order> perform(
int playerId,
GameState state,
List<UUID> entities
);

}
```

Basic-TBS

- Rules now apply to **0** or more Entities

Basic-TBS

- Rules now apply to **0** or more Entities
- **Remaining** entities considered by lower rules

Basic-TBS

- Rules now apply to **0** or more Entities
- **Remaining** entities considered by lower rules
- Orders generated by rules are **executed** by the rule

Basic-TBS

- Rules now apply to **0** or more Entities
- **Remaining** entities considered by lower rules
- Orders generated by rules are **executed** by the rule
    - **Simulating** their effects for other rules

Basic-TBS

- Rules now apply to **0** or more Entities
- **Remaining** entities considered by lower rules
- Orders generated by rules are **executed** by the rule
  - **Simulating** their effects for other rules
  - Allows **lower** rules to make **informed** decisions

Basic-TBS

- Rules now apply to **0** or more Entities
- **Remaining** entities considered by lower rules
- Orders generated by rules are **executed** by the rule
  - **Simulating** their effects for other rules
  - Allows **lower** rules to make **informed** decisions
  - Is why **non-determinism** is ill-advised

Built-Ins

- Many rules are provided

Built-Ins

- Many rules are provided
- Some complications:

Built-Ins

- Many rules are provided
- Some complications:
    - Rules deal with **EntityTypes** and **Actions**

Built-Ins

- Many rules are provided
- Some complications:
  - Rules deal with **EntityTypes** and **Actions**
  - When I write the rules, you haven't **written** them yet

Built-Ins

- Many rules are provided
- Some complications:
  - Rules deal with **EntityTypes** and **Actions**
  - When I write the rules, you haven't **written** them yet
  - Luckily they are **dynamically** built at **runtime**

Built-Ins

- Many rules are provided
- Some complications:
  - Rules deal with **EntityTypes** and **Actions**
  - When I write the rules, you haven't **written** them yet
  - Luckily they are **dynamically** built at **runtime**
- I suppose we should list the included rules

Built-Ins

- Many rules are provided
- Some complications:
  - Rules deal with **EntityTypes** and **Actions**
  - When I write the rules, you haven't **written** them yet
  - Luckily they are **dynamically** built at **runtime**
- I suppose we should list the included rules
- And how to **use** them

Built-Ins

AttackMeleeRule	AttackRangedClosestRule
AttackRangedMostDamage-dRule	EnsureEntityRule
Module	NoopProductionRule
RandomRule	Filter
	RunAwayRule
RunTowardsRule	RunTowardsResource
UseActionOnEntity	UseActionOnResource

Some rules are similar - Will cover them together

Rule Types

- Two main types

Rule Types

- Two main types
    - ProductionRule

Rule Types

- Two main types
  - ProductionRule
  - PerEntityRule *implements* ProductionRule

Rule Types

- Two main types
  - ProductionRule
  - PerEntityRule *implements* ProductionRule
- Depends what you need to do in the rule

Rule Types

- Two main types
  - ProductionRule
  - PerEntityRule *implements* ProductionRule
- Depends what you need to do in the rule
- PerEntityRule is simpler

Rule Types

- Two main types
  - ProductionRule
  - PerEntityRule *implements* ProductionRule
- Depends what you need to do in the rule
- PerEntityRule is simpler
  - Executes the orders **automatically**

Rule Types

- Two main types
  - ProductionRule
  - PerEntityRule *implements* ProductionRule
- Depends what you need to do in the rule
- PerEntityRule is simpler
  - Executes the orders **automatically**
  - What a **single** Entity does

Rule Types

- Two main types
  - ProductionRule
  - PerEntityRule *implements* ProductionRule
- Depends what you need to do in the rule
- PerEntityRule is simpler
  - Executes the orders **automatically**
  - What a **single** Entity does
  - Removes Entity from consideration for you

Attack Rules

- PerEntityRule

Attack Rules

- PerEntityRule
- AttackMeleeRule, AttackRangedClosestRule,AttackRangedMostDamagedRule

Attack Rules

- PerEntityRule
- AttackMeleeRule, AttackRangedClosestRule,AttackRangedMostDamagedRule
- Melee rule will also **move** unit towards target

Attack Rules

- PerEntityRule
- AttackMeleeRule, AttackRangedClosestRule,AttackRangedMostDamagedRule
- Melee rule will also **move** unit towards target
- Be careful:

Attack Rules

- PerEntityRule
- AttackMeleeRule, AttackRangedClosestRule,AttackRangedMostDamagedRule
- Melee rule will also **move** unit towards target
- Be careful:
  - These rules do **not** check

Attack Rules

- PerEntityRule
- AttackMeleeRule, AttackRangedClosestRule,AttackRangedMostDamagedRule
- Melee rule will also **move** unit towards target
- Be careful:
  - These rules do **not** check
  - They can issue an **invalid** order

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity
- Simple:

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity
- Simple:
  - Counts how many Product we have

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity
- Simple:
  - Counts how many Product we have
  - Finds Producer for each missing Product

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity
- Simple:
  - Counts how many Product we have
  - Finds Producer for each missing Product
  - Issues order for Producer to build Product

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity
- Simple:
  - Counts how many Product we have
  - Finds Producer for each missing Product
  - Issues order for Producer to build Product
- Supports abstract types:

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity
- Simple:
  - Counts how many Product we have
  - Finds Producer for each missing Product
  - Issues order for Producer to build Product
- Supports abstract types:
- EnsureEntity[abstract_civilian:abstract_town:1]

EnsureEntity

Example

EnsureEntity[blue_town:blue_civilian:3]

- Producer, Product, Quantity
- Simple:
  - Counts how many Product we have
  - Finds Producer for each missing Product
  - Issues order for Producer to build Product
- Supports abstract types:
- EnsureEntity[abstract_civilian:abstract_town:1]
- Uses BuildOrder.

Run Rules

Example

RunTowards[0.0]

RunAway[0.5]

RunTowardsResource[gold]

- Causes Entity to travel

Run Rules

Example

RunTowards[0.0]

RunAway[0.5]

RunTowardsResource[gold]

- Causes Entity to travel
- **To** or **from** something

Use Rules

Example

UseActionOnResource[BuildOnResource[gold_mine:gold]:gold]

- Causes Entity to use an Action on either:

Use Rules

Example

UseActionOnResource[BuildOnResource[gold_mine:gold]:gold]

- Causes Entity to use an Action on either:
  - Another Entity

Use Rules

Example

UseActionOnResource[BuildOnResource[gold_mine:gold]:gold]

- Causes Entity to use an Action on either:
  - Another Entity
  - A resource

Use Rules

Example

UseActionOnResource[BuildOnResource[gold_mine:gold]:gold]

- Causes Entity to use an Action on either:
  - Another Entity
  - A resource
- Great for building mines

Use Rules

Example

UseActionOnResource[BuildOnResource[gold_mine:gold]:gold]

- Causes Entity to use an Action on either:
  - Another Entity
  - A resource
- Great for building mines
- Great for supporting custom actions.

Use Rules

Example

UseActionOnResource[BuildOnResource[gold_mine:gold]:gold]

- Causes Entity to use an Action on either:
  - Another Entity
  - A resource
- Great for building mines
- Great for supporting custom actions.
- Attack actions could have been written with this

Use Rules

Example

UseActionOnResource[BuildOnResource[gold_mine:gold]:gold]

- Causes Entity to use an Action on either:
  - Another Entity
  - A resource
- Great for building mines
- Great for supporting custom actions.
- Attack actions could have been written with this
- But they predate this

Filter Rules

Example

Filter[AttackMelee:abstract_knight]

- PerEntityRule

Filter Rules

Example

Filter[AttackMelee:abstract_knight]

- PerEntityRule
- PerEntityRule usually consults **every** Entity

Filter Rules

Example

Filter[AttackMelee:abstract_knight]

- PerEntityRule
- PerEntityRule usually consults **every** Entity
- This doesn't **usually** make sense

Filter Rules

Example

Filter[AttackMelee:abstract_knight]

- PerEntityRule
- PerEntityRule usually consults **every** Entity
- This doesn't **usually** make sense
- Rather than write **conditions** on other rules

Filter Rules

Example

Filter[AttackMelee:abstract_knight]

- PerEntityRule
- PerEntityRule usually consults **every** Entity
- This doesn't **usually** make sense
- Rather than write **conditions** on other rules
- **Wrap** rules with this condition

Filter Rules

Example

Filter[AttackMelee:abstract_knight]

- PerEntityRule
- PerEntityRule usually consults **every** Entity
- This doesn't **usually** make sense
- Rather than write **conditions** on other rules
- **Wrap** rules with this condition
- Takes 1 or more type as an array

Custom Rules

- These are largely designed for civ style games

Custom Rules

- These are largely designed for civ style games
- Great news if you have made one

Custom Rules

- These are largely designed for civ style games
- Great news if you have made one
- Less than great news if your game is radically different

Custom Rules

- These are largely designed for civ style games
- Great news if you have made one
- Less than great news if your game is radically different
- You'll need to **provide** new rules

Custom Rules

- These are largely designed for civ style games
- Great news if you have made one
- Less than great news if your game is radically different
- You'll need to **provide** new rules
    - Same way you did for Actions

Custom Rules

- These are largely designed for civ style games
- Great news if you have made one
- Less than great news if your game is radically different
- You'll need to **provide** new rules
  - Same way you did for Actions
  - Dynamically scanned at runtime

Production Rule Agents

”RangedRush”:”PRA[EnsureBase,EnsureWorker,EnsureArchery,BuildGoldMine,BuildWoodMine,EnsureEntity[abstract_civilian:farm:3],TravelToGold,TravelToWood,EnsureArcher,ArcherAttack,ArcherChase,WorkerEvade]”,

Production Rule Agents

- That was a lot in one line

Production Rule Agents

- That was a lot in one line
- **Sorry**, but that is how it is

Production Rule Agents

- That was a lot in one line
- **Sorry**, but that is how it is
- These do tend to be quite in-flexible.

Production Rule Agents

- That was a lot in one line
- **Sorry**, but that is how it is
- These do tend to be quite in-flexible.
- More on that later

Genetic Algorithms

Genetic Algorithms

- Powerful algorithms for variety of uses

Genetic Algorithms

- Powerful algorithms for variety of uses
- Can even **play** games

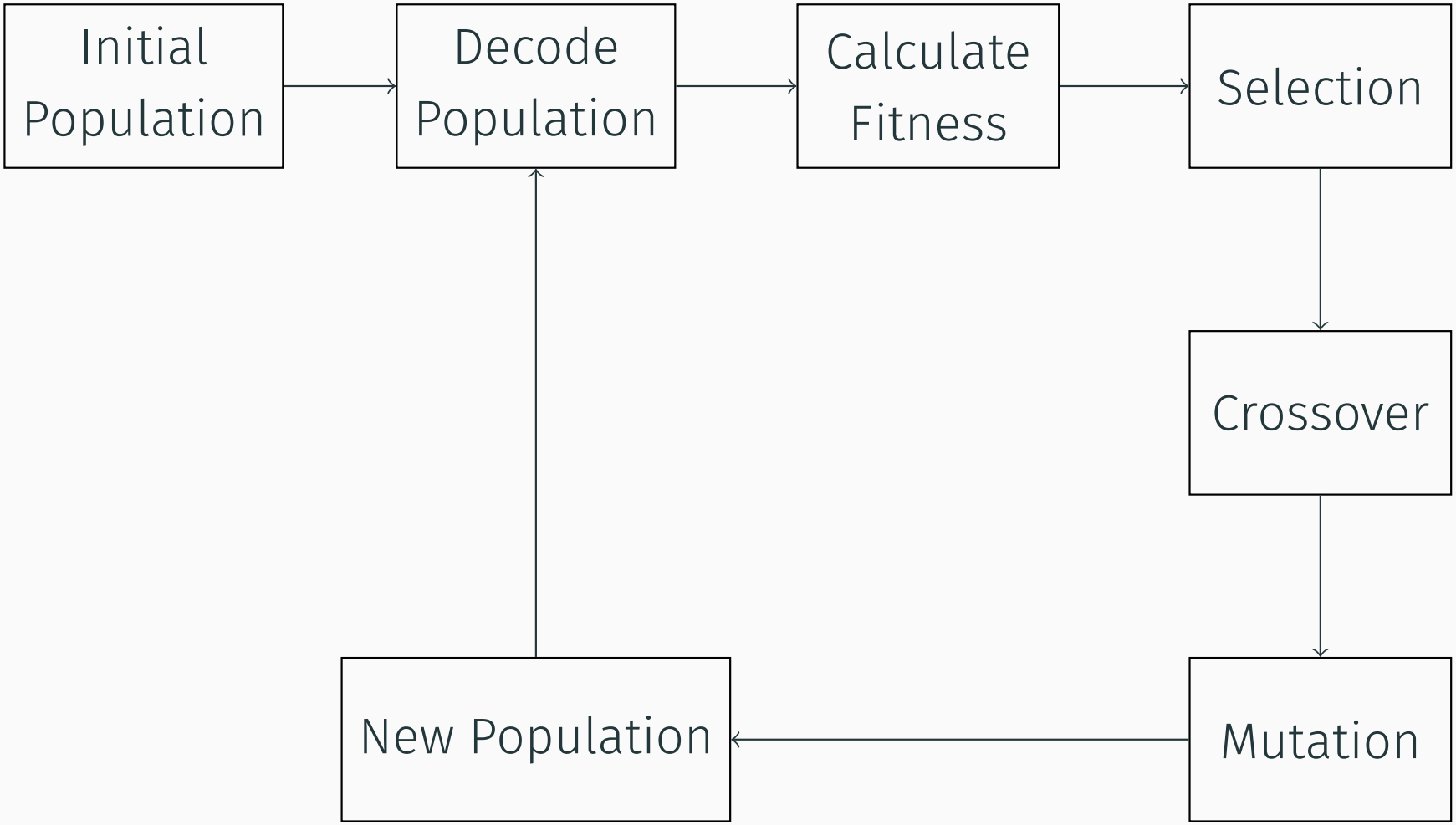
Genetic Algorithms

- Powerful algorithms for variety of uses
- Can even **play** games
- Apologies to those that have encountered GA's

Genetic Algorithms

- Powerful algorithms for variety of uses
- Can even **play** games
- Apologies to those that have encountered GA's
- Even more apologies to those that have encountered RHEA's

Genetic Algorithms



Genetic Algorithms

- You **all** have some experience using these

Genetic Algorithms

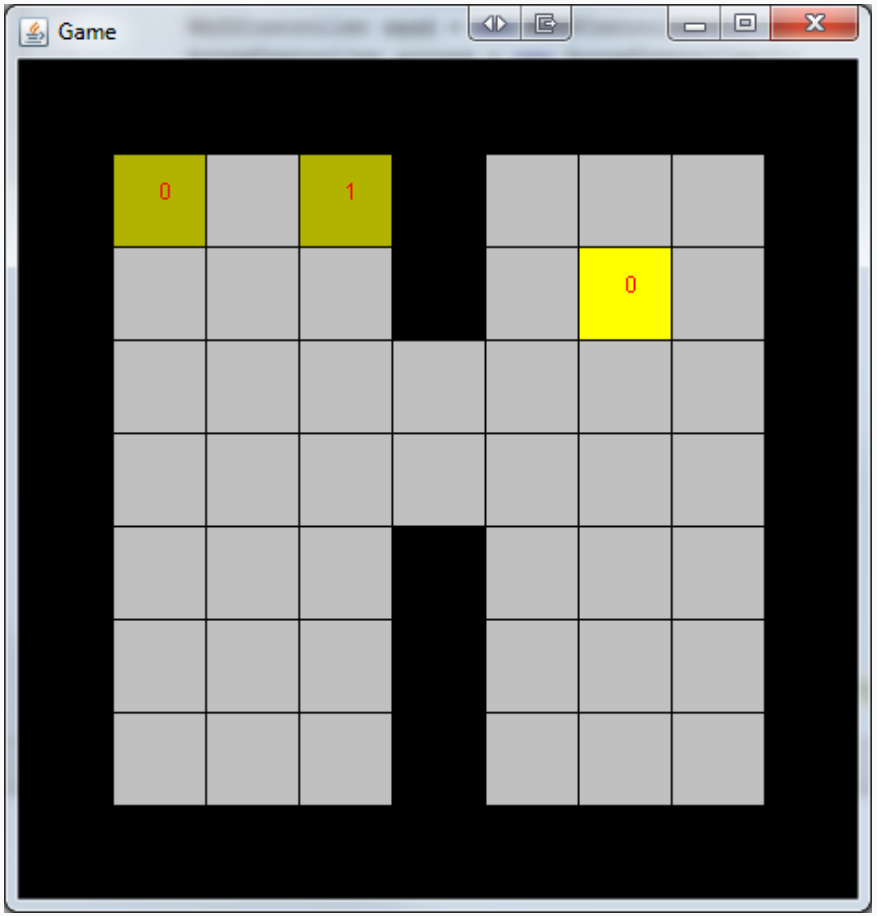
- You **all** have some experience using these
- Asteroids assignment had one built in

Genetic Algorithms

- You **all** have some experience using these
- Asteroids assignment had one built in
- But how to play a game with a GA?

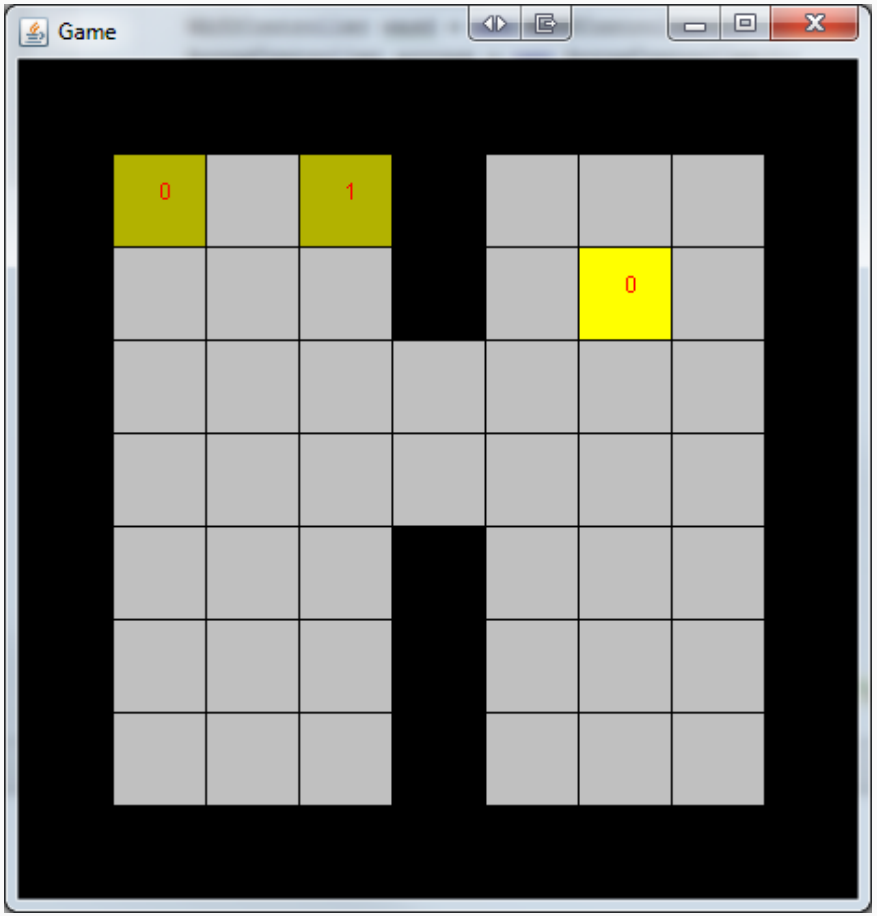
Simpler Games

- Consider this game



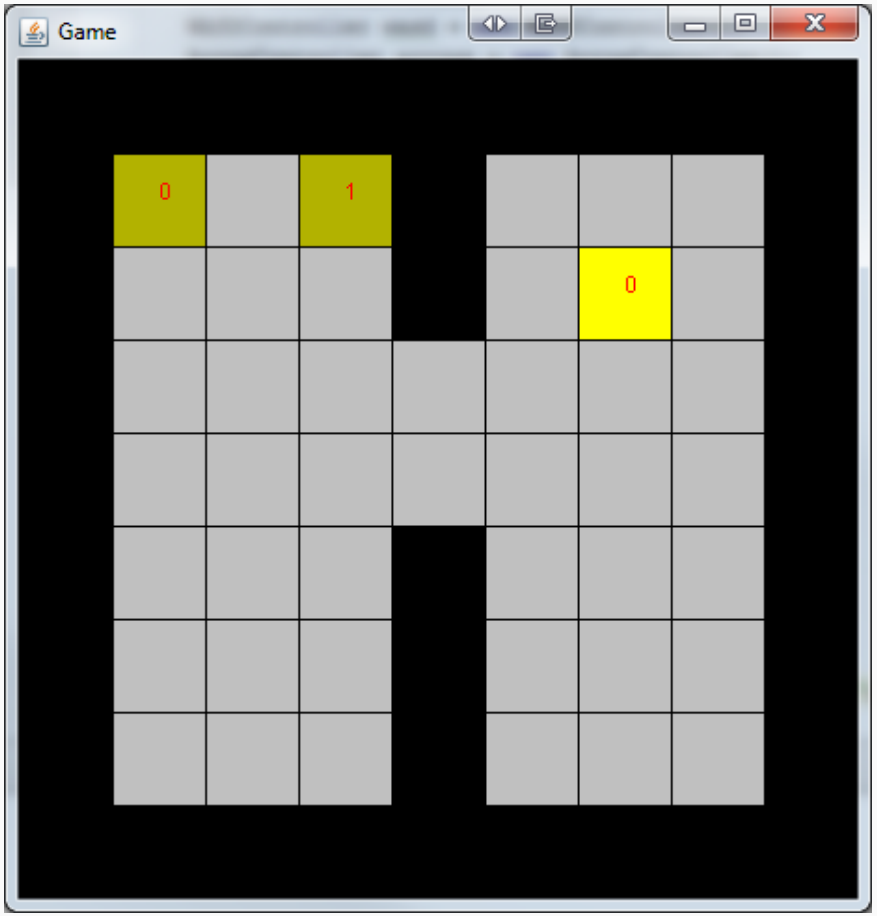
Simpler Games

- Consider this game
- More like a maze



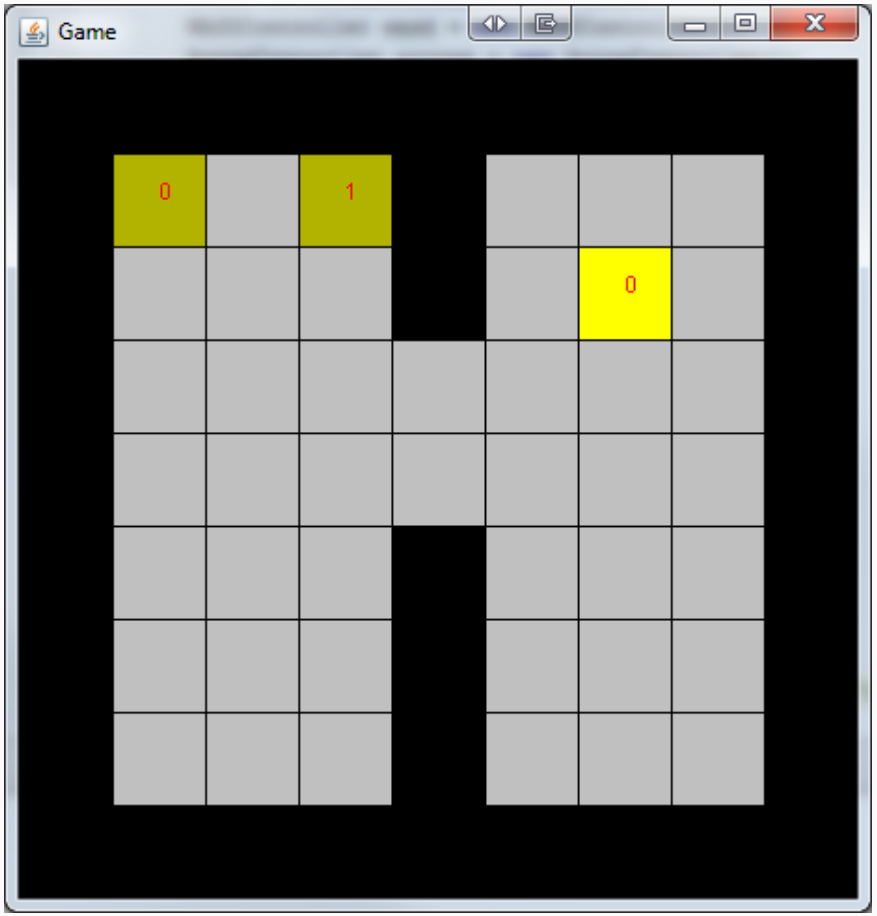
Simpler Games

- Consider this game
- More like a maze
- Get each Agent to the goal



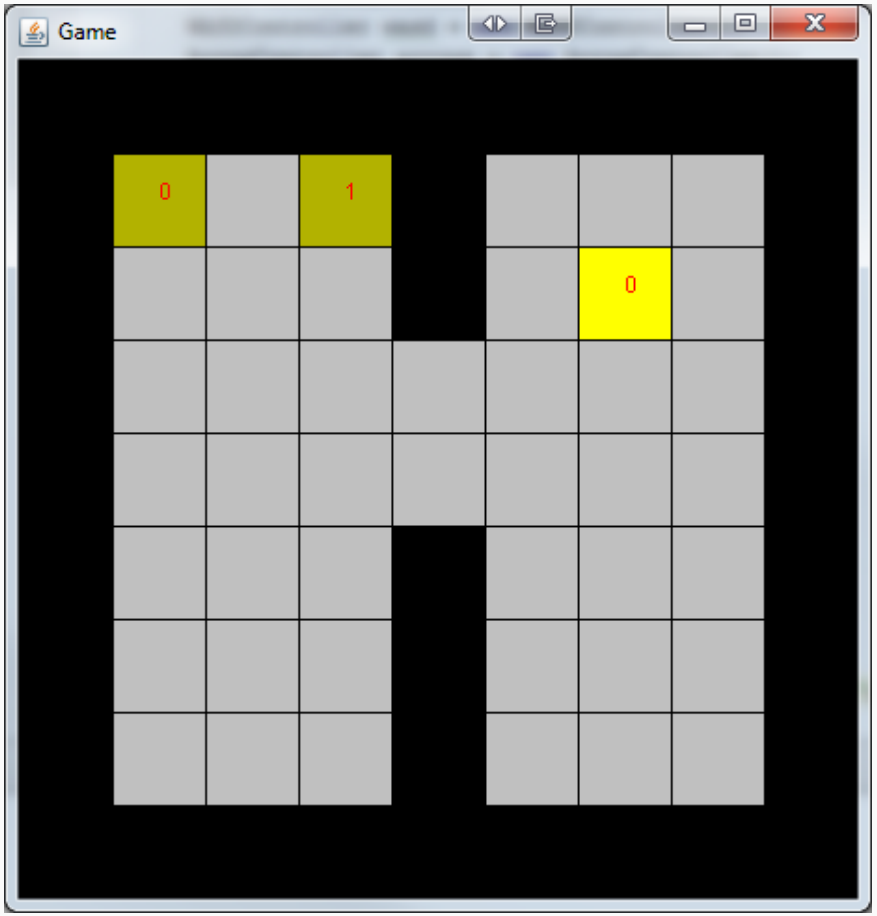
Simpler Games

- Consider this game
- More like a maze
- Get each Agent to the goal
- 5 possible moves



Simpler Games

- Consider this game
- More like a maze
- Get each Agent to the goal
- 5 possible moves
- Controller returns a single move per turn



GA for Simple Game

- Given a time budget

GA for Simple Game

- Given a time budget
- Spend it **evolving** “plans”

GA for Simple Game

- Given a time budget
- Spend it **evolving** “plans”
- Plan is a **sequence** of possible actions

GA for Simple Game

- Given a time budget
- Spend it **evolving** “plans”
- Plan is a **sequence** of possible actions
- Simulate the plan and evaluate resultant state for score

GA for Simple Game

- Given a time budget
- Spend it **evolving** “plans”
- Plan is a **sequence** of possible actions
- Simulate the plan and evaluate resultant state for score
- Sound like a possible GA?

GA for Simple Game

- Given a time budget
- Spend it **evolving** “plans”
- Plan is a **sequence** of possible actions
- Simulate the plan and evaluate resultant state for score
- Sound like a possible GA?
- This is called a Rolling Horizon Evolutionary Algorithm

Boosting the horizon

- Length of the sequence increases parameters for GA

Boosting the horizon

- Length of the sequence increases parameters for GA
- RHEA is quite **jerky** in games

Boosting the horizon

- Length of the sequence increases parameters for GA
- RHEA is quite **jerky** in games
- Macro Actions can sometimes solve this

Boosting the horizon

- Length of the sequence increases parameters for GA
- RHEA is quite **jerky** in games
- Macro Actions can sometimes solve this
- Locking the agent to consider each move *N* times

Boosting the horizon

- Length of the sequence increases parameters for GA
- RHEA is quite **jerky** in games
- Macro Actions can sometimes solve this
- Locking the agent to consider each move *N* times
- Means you can think for *N* turns

Boosting the horizon

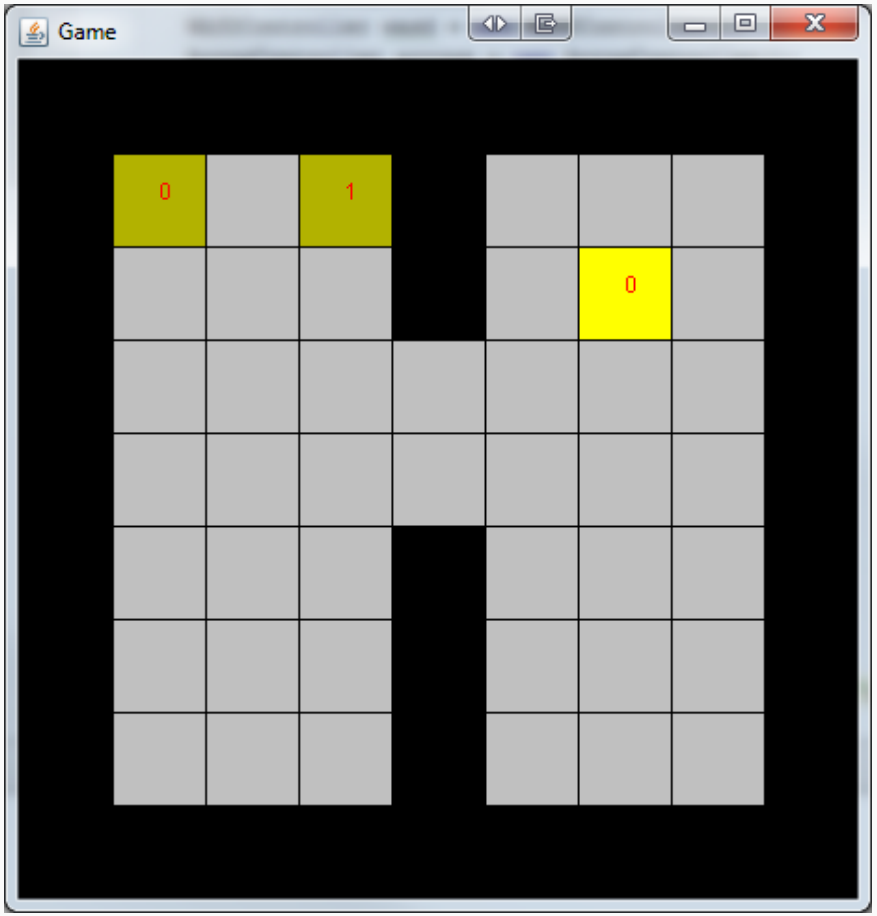
- Length of the sequence increases parameters for GA
- RHEA is quite **jerky** in games
- Macro Actions can sometimes solve this
- Locking the agent to consider each move *N* times
- Means you can think for *N* turns
- Works great in real-time engines like PTSP

Boosting the horizon

- Length of the sequence increases parameters for GA
- RHEA is quite **jerky** in games
- Macro Actions can sometimes solve this
- Locking the agent to consider each move *N* times
- Means you can think for *N* turns
- Works great in real-time engines like PTSP
- Getting *N* wrong means poor performance

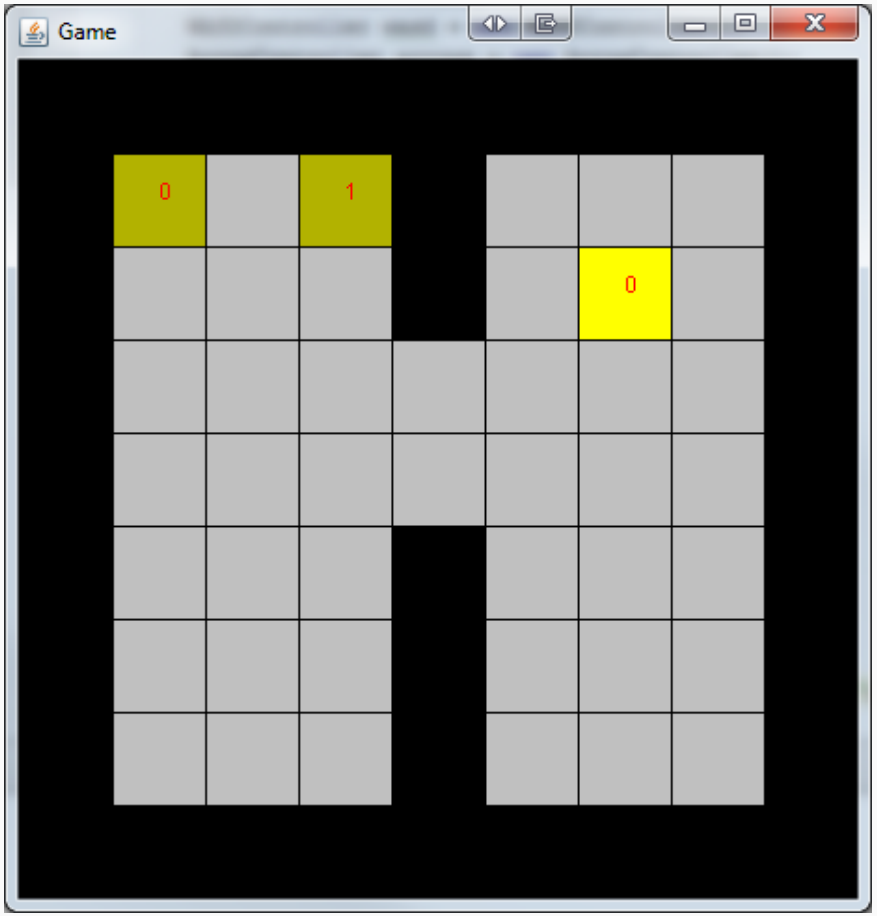
Problem with Macro Actions

- What if N is 2?



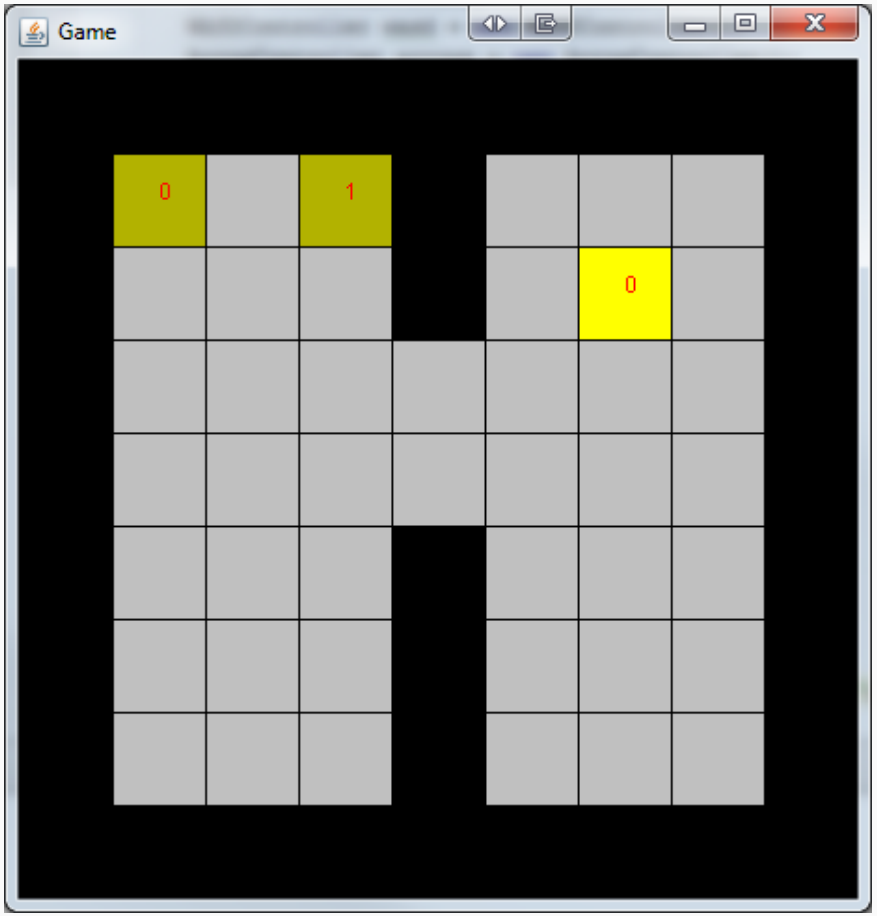
Problem with Macro Actions

- What if N is 2?
- Can we reach the Goal?



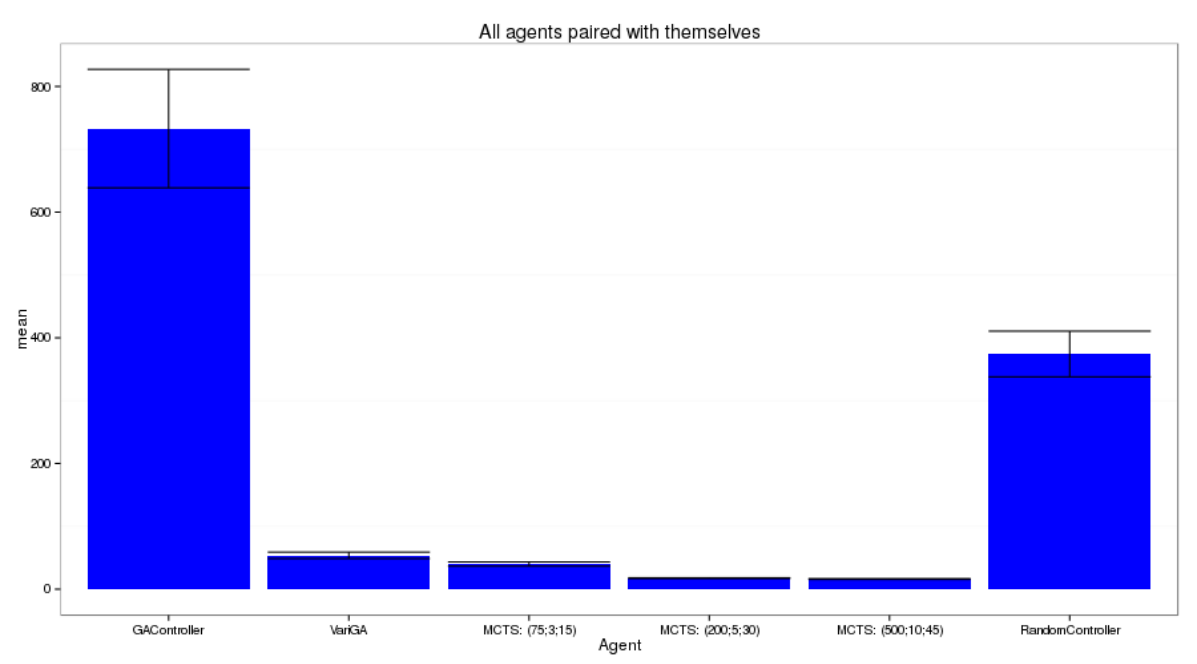
Problem with Macro Actions

- What if *N* is 2?
- Can we reach the Goal?
- MacroActionGA is poor at discrete boards



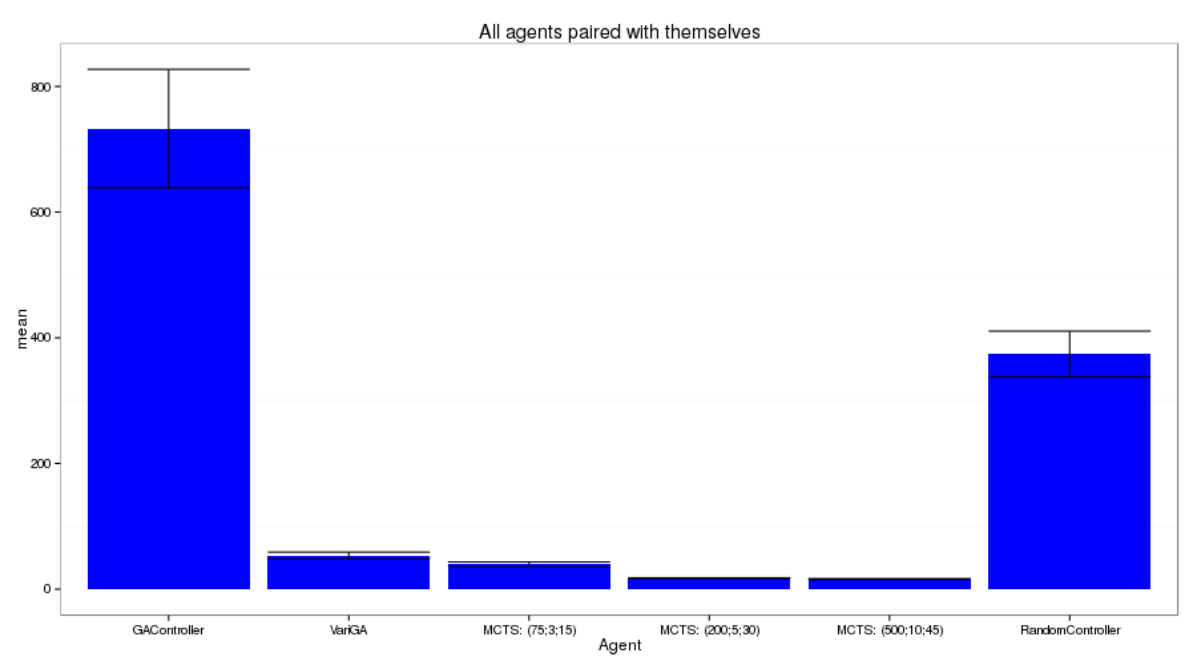
Solution

- What if *N* wasn't fixed?



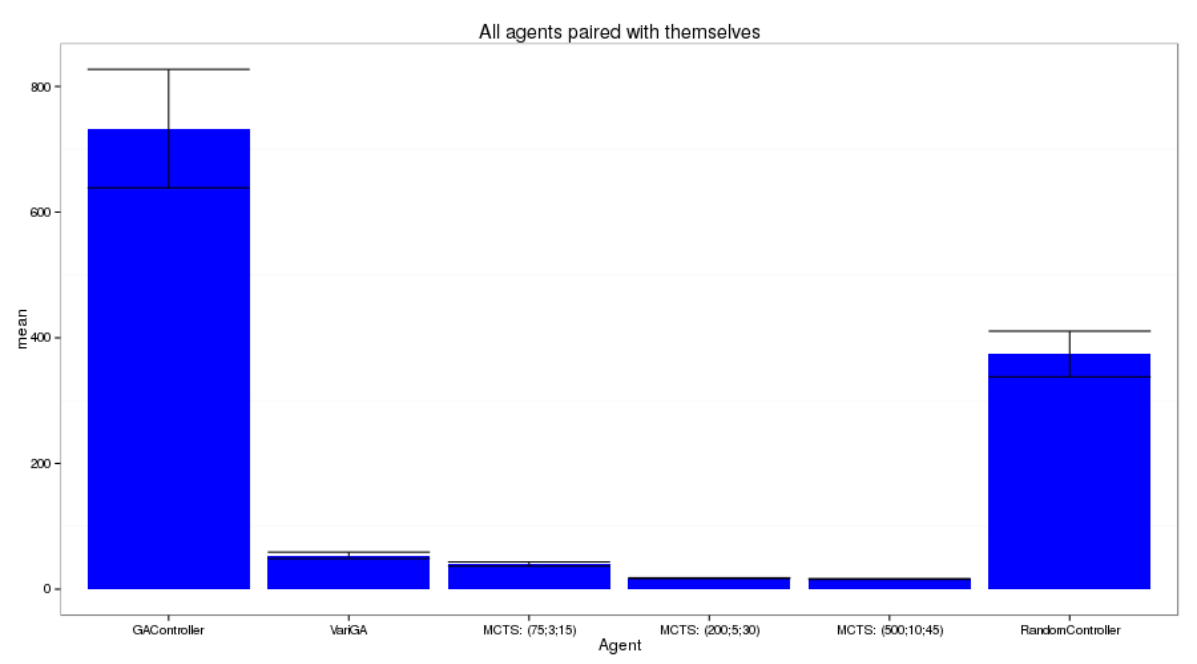
Solution

- What if *N* wasn't fixed?
- What if *N* wasn't the same for each action?



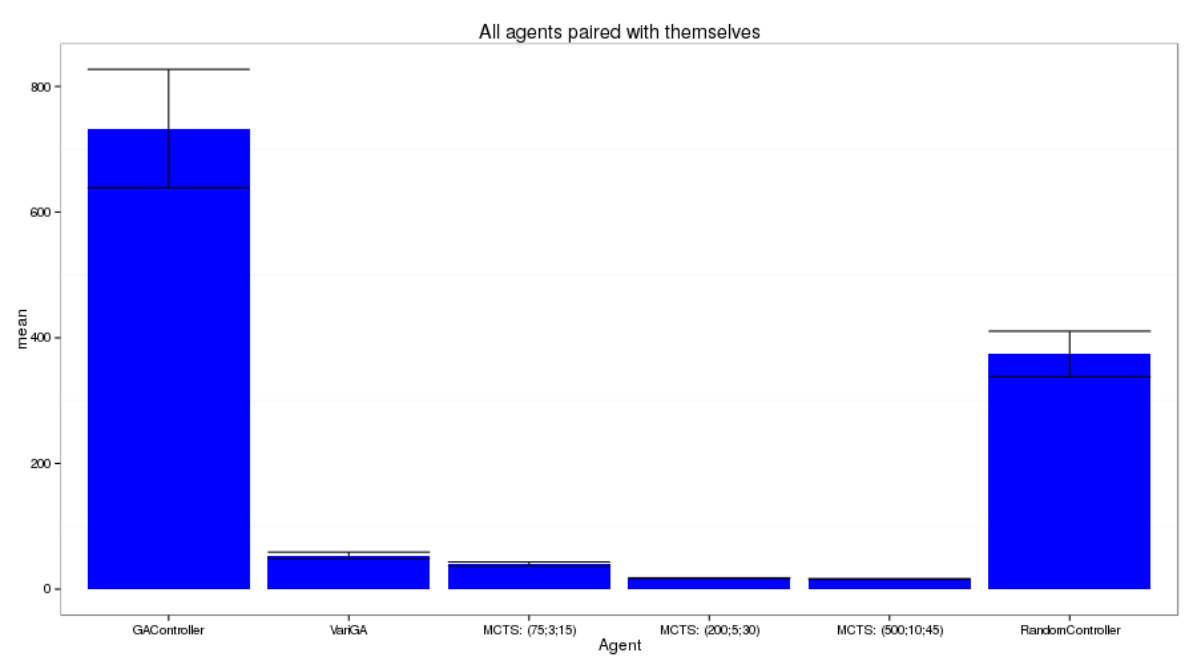
Solution

- What if *N* wasn't fixed?
- What if *N* wasn't the same for each action?
- Learn the values for *N* as well as the actions



Solution

- What if *N* wasn't fixed?
- What if *N* wasn't the same for each action?
- Learn the values for *N* as well as the actions
- Include them in the GA



More complex games

- This is great for simple games

More complex games

- This is great for simple games
- More complex games are too large

More complex games

- This is great for simple games
- More complex games are too large
- Action space in our game is huge!

More complex games

- This is great for simple games
- More complex games are too large
- Action space in our game is huge!
- Macro actions don’t make sense either!

More complex games

- This is great for simple games
- More complex games are too large
- Action space in our game is huge!
- Macro actions don’t make sense either!
- Need something **higher** level

Strategy Search

- Instead of choosing actions

Strategy Search

- Instead of choosing actions
- Choose between **strategies**

Strategy Search

- Instead of choosing actions
- Choose between **strategies**
- But where will we get those?

Strategy Search

```
Map<EntityType, List<ProductionRuleAgent>> rules;
```

- Choose between PRA's for each **EntityType**

Strategy Search

```
Map<EntityType, List<ProductionRuleAgent>> rules;
```

- Choose between PRA's for each **EntityType**
- Why not per Entity?

Variable Length Macro Action GA

”MedievalGA”:”VLMAGA[1000:10:EandM:abstract_civilian,abstract_town,abstract_knightery,abstract_archery,abstract_knight,abstract_archer:noopRule/ResourceBuilder/BuildBase/BuildMilitary,noopRule/BuildWorker,noopRule/BuildKnights,noopRule/BuildArchers,KnightAttack,ArcherAttack:noop:RangedRush]”

- Will do automatic single actions first

Variable Length Macro Action GA

”MedievalGA”:”VLMAGA[1000:10:EandM:abstract_civilian,abstract_town,abstract_knightery,abstract_archery,abstract_knight,abstract_archer:noopRule/ResourceBuilder/BuildBase/BuildMilitary,noopRule/BuildWorker,noopRule/BuildKnights,noopRule/BuildArchers,KnightAttack,ArcherAttack:noop:RangedRush]”

- Will do automatic single actions first
- Then will follow its learned policy

Variable Length Macro Action GA

”MedievalGA”:”VLMAGA[1000:10:EandM:abstract_civilian,abstract_town,abstract_knightery,abstract_archery,abstract_knight,abstract_archer:noopRule/ResourceBuilder/BuildBase/BuildMilitary,noopRule/BuildWorker,noopRule/BuildKnights,noopRule/BuildArchers,KnightAttack,ArcherAttack:noop:RangedRush]”

- Will do automatic single actions first
- Then will follow its learned policy
- Then will use the fallback agent for the rest

Problems

Controller	ops/min	Error
random	512.840	13.846
noop	509.314	5.730
RangedRush	273.362	2.345
MixedRush	284.875	7.453
MedievalGA	3.538	2.286

- An “op” is building a full game and playing it to the end
- MedievalGA is the VLMAGA
- I wish that were an error for MedievalGA