Rinascimento

Optimising Statistical Forward Planning Agents for Playing Splendor

Ivan Bravi
Diego Perez-Liebana
Simon Lucas

Queen Mary University of London

Jialin Liu

SUSTech Southern University of Science and Technology

Keywords

Splendor: the real board game*

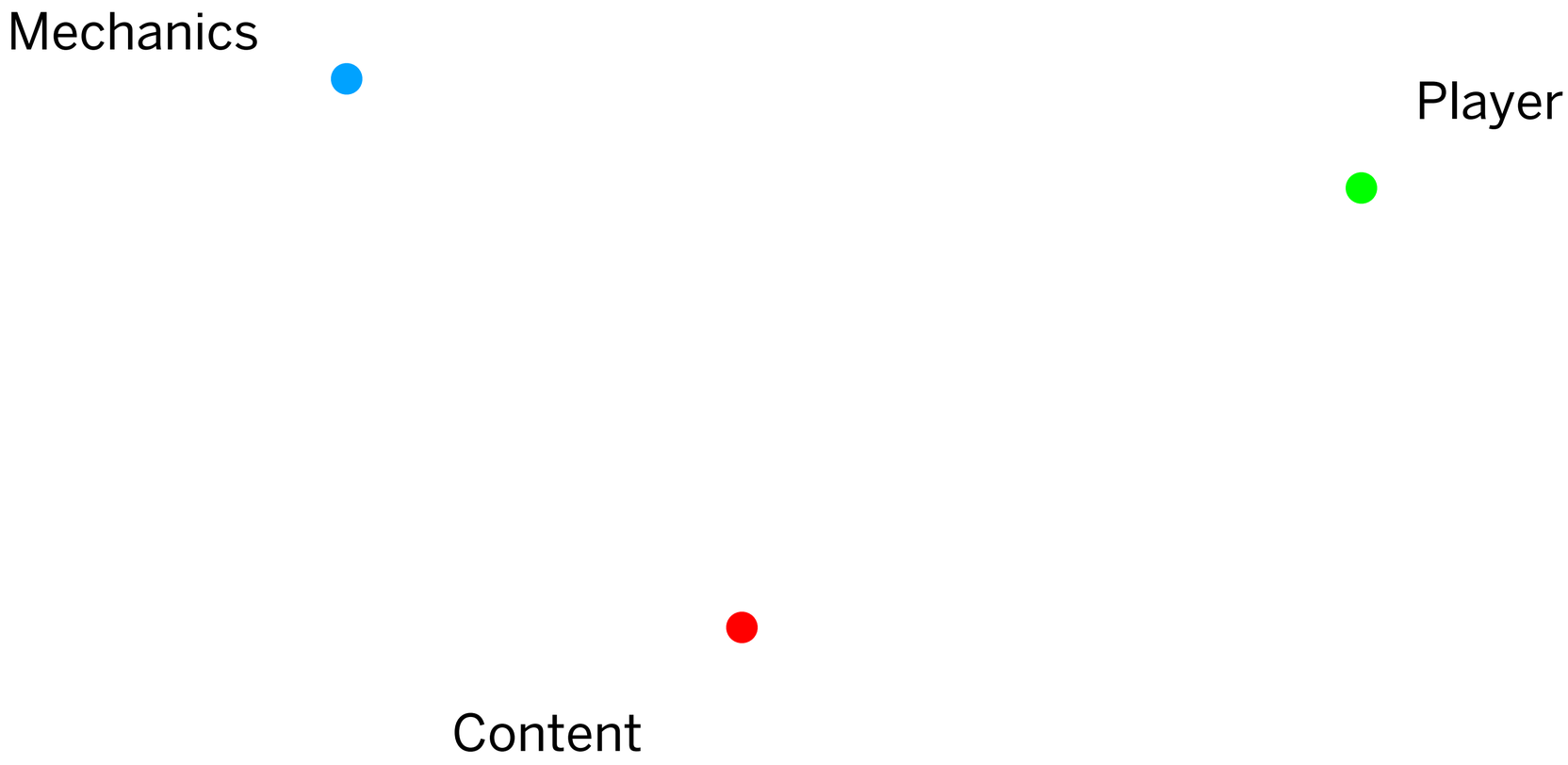
ℜinascimento: the framework implementing the game(s)

Statistical Forward Planning (SFP): AI game-playing algorithms that use a forward-model to simulate future game states (e.g. MCTS, RHEA)

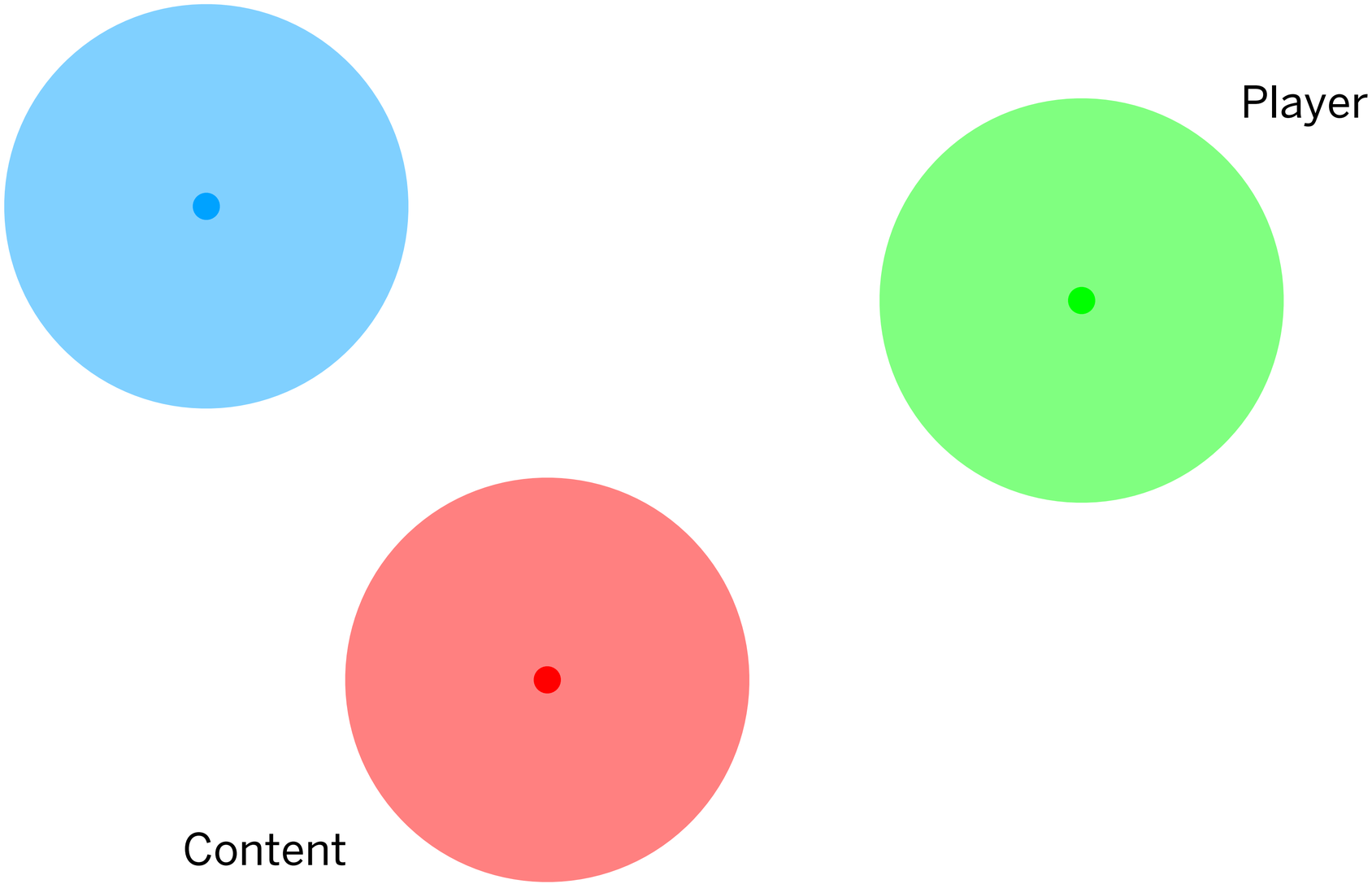
Hyper-parameter: the parameters governing the behaviour of the algorithm

**designed by Marc André and published by Space Cowboys*

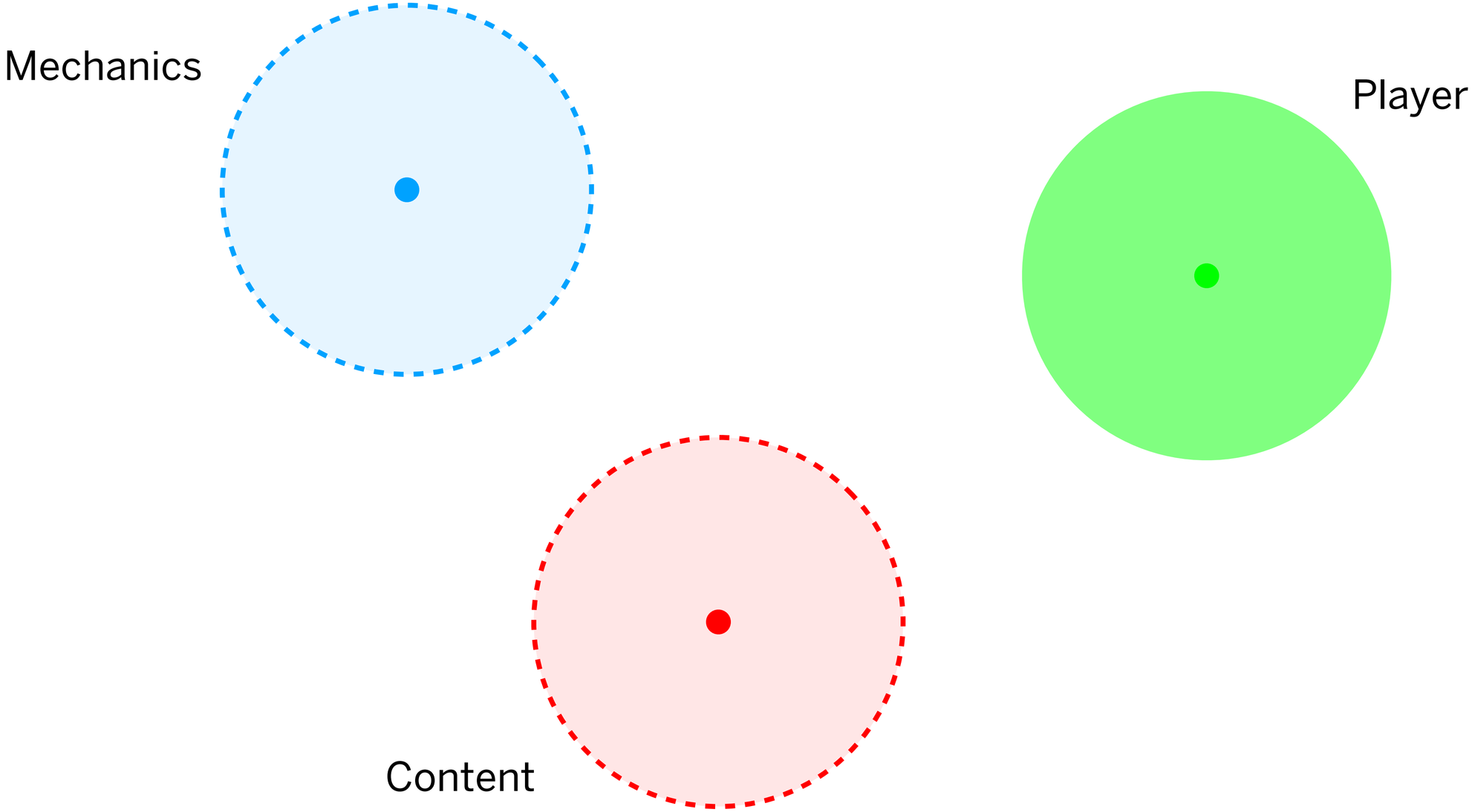
Purpose



Purpose



Purpose



Splendor

The Game

4-Player Setup

- 5 random nobles
- 3 decks
- 4 face-up cards per deck
- 7 common tokens per type
- 5 joker tokens

Actions

Pick common tokens	• up to 3 of 3 different types • 2 of the same type (only if there are at least 4 tokens)
Reserve card	• from the table + 1 joker • from a deck (hidden to the other players) + 1 joker
Buy card	• from the table • from your reserved ones

Why Splendor

Simple but Complex

Simplicity

clear discrete game state representation

simple actions

instant game events

Complexity

long term implications of early-game actions

limited game duration

hard to master

partial observability

multi-player

Rinascimento

The Game AI Framework

Parameters

#players	4	#decks *	3
#token types*	5	#extra nobles*	1
#coins per token	7	pick different #token types	3
#gold tokens	5	pick different #tokens	1
max #tokens	10	pick same #coins	2
end game prestige	15	pick same min #coins	4
#face-up cards	4	reserve max #cards	3

**requires Procedural Content Generation*

Dimensionality

14 dimensions

Classic 4 players Splendor

[4 , 5 , 7 , 5 , 3 , 4 , 1 , 10 , 15 , 3 , 1 , 2 , 4 , 3]

Action Space

Usually

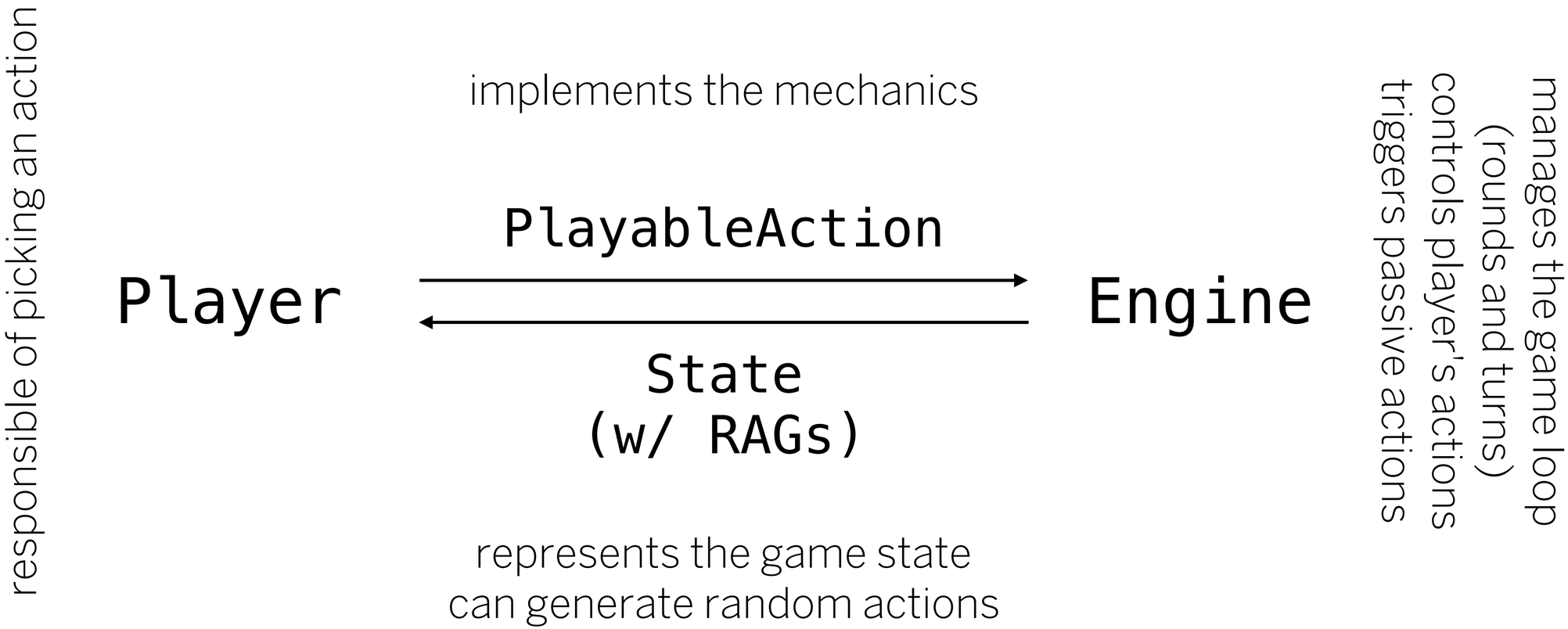
- count actions theoretical upper bound
- provide explicit action space

Action Space

Random Action Generator (RAG)

- each action type has a RAG that can generate random *legal* action
- it can be seeded to *control* the randomness
- samples the legal action space

Architecture



Performance

Running 10,000 games of 4 random players*

Stalemate frequency	1,410/10,000
Speed	~1.74M states/s
Average full game duration	0.44 ms

*6yo 2.3 GHz Intel Core i7

Statistical Forward Planning

The Game-playing AI

Algorithms

Basic

Random

One-Step Look Ahead

Advanced

Seeded Rolling Horizon (SRH)

Branching Mutation Rolling Horizon (BMRH)

Monte Carlo Tree Search (MCTS) + Iterative Widening

Seeded RH

Classic Rolling Horizon

a sequence of *actions* [0 , 1 , 0 , 4 , 2]

Seeded Rolling Horizon

a sequence of *seeds* [0.127 , 0.9852 , 0.2039 , 0.51 , 0.2547]

Branching Mutation RH

Classic Rolling Horizon

one-shot: [0 , 1 , 0 , 4 , 2]

$\mathfrak{R}$

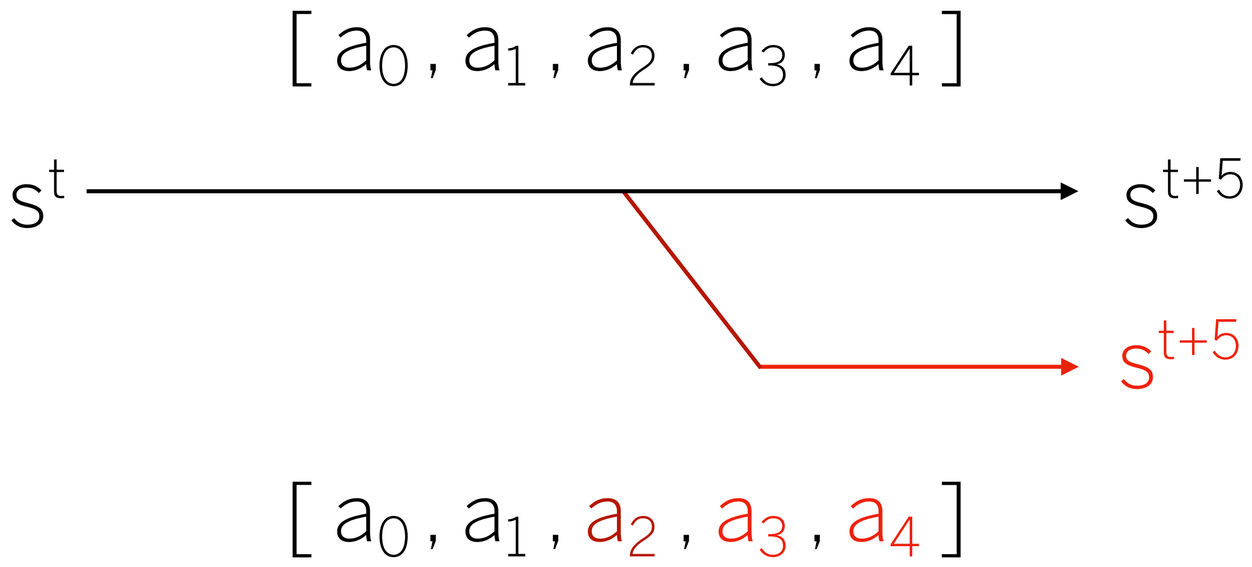
$[a_0 , - , - , - , -]$ $\quad a_0 = rnd(s^t_{\{\}})$

$[a_0 , a_1 , - , - , -]$ $\quad a_1 = rnd(s^t_{\{a_0\}})$

...

$[a_0 , a_1 , a_2 , a_3 , a_4]$ $\quad a_4 = rnd(s^t_{\{a_0 a_1 a_2 a_3 a_4\}})$

Branching Mutation RH



Hyper-parameters

BMRH

Symbol	Type	Description
l	integer	sequence length
n	integer	sequences evaluated
usb	boolean	if it uses shift buffer
mo	boolean	if it has to mutate once
ms	integer	mutation type
dcy	double	probability of exponential decay
μ	double	mean of the gaussian mutation point
σ	double	std dev of the gaussian mutation point

SRH

Symbol	Type	Description
l	integer	sequence length
n	integer	sequences evaluated
usb	boolean	if it uses shift buffer
mo	boolean	if it has to mutate once
mr	double	mutation probability

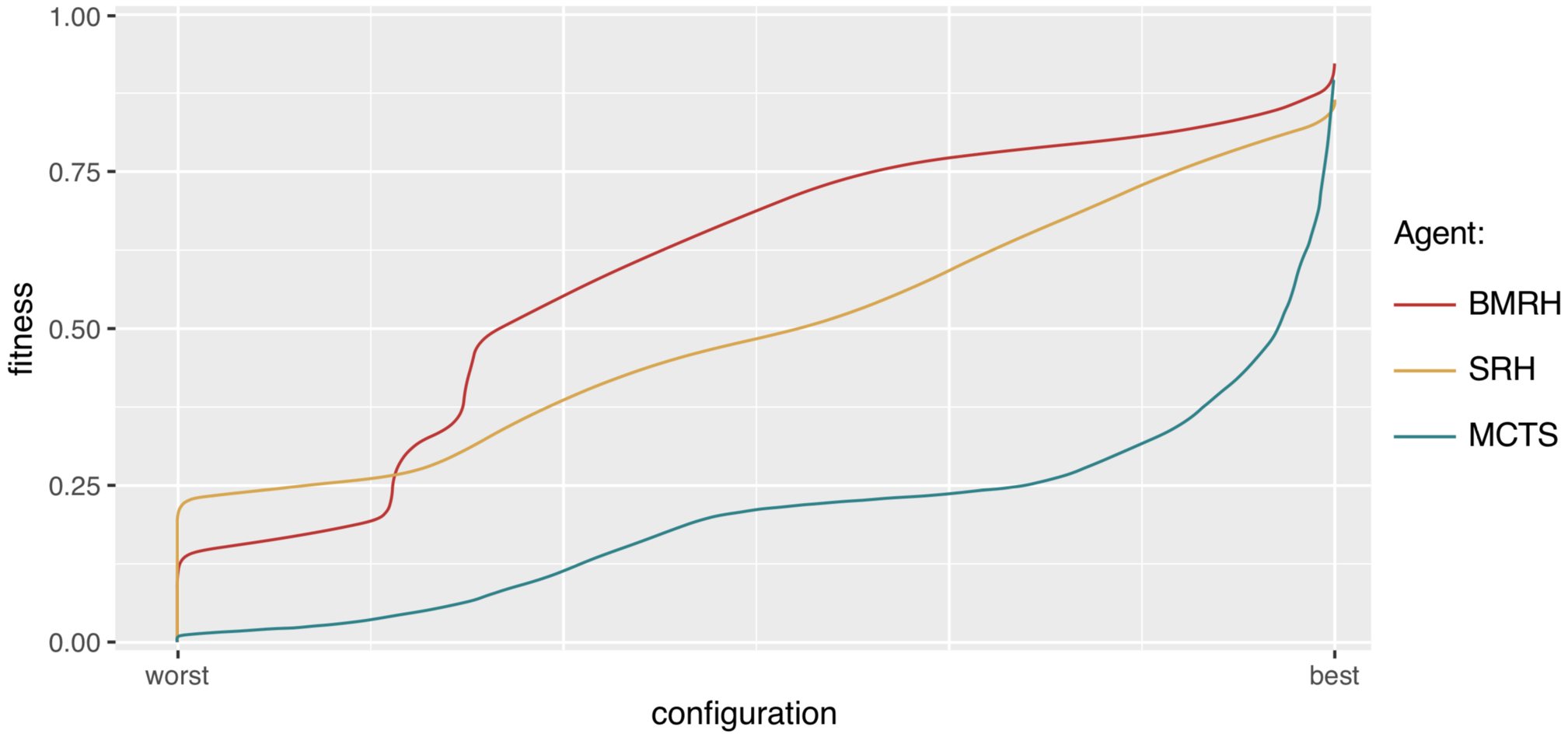
MCTS

Symbol	Type	Description
d	integer	max depth reached by the tree or the rollout
c	double	exploration constant of UCB
e	double	ϵ of UCB
ep	double	probability of further expanding the node
ps	integer	number of actions sampled during expansion
rt	integer	recommendation type

Experiments

to the test

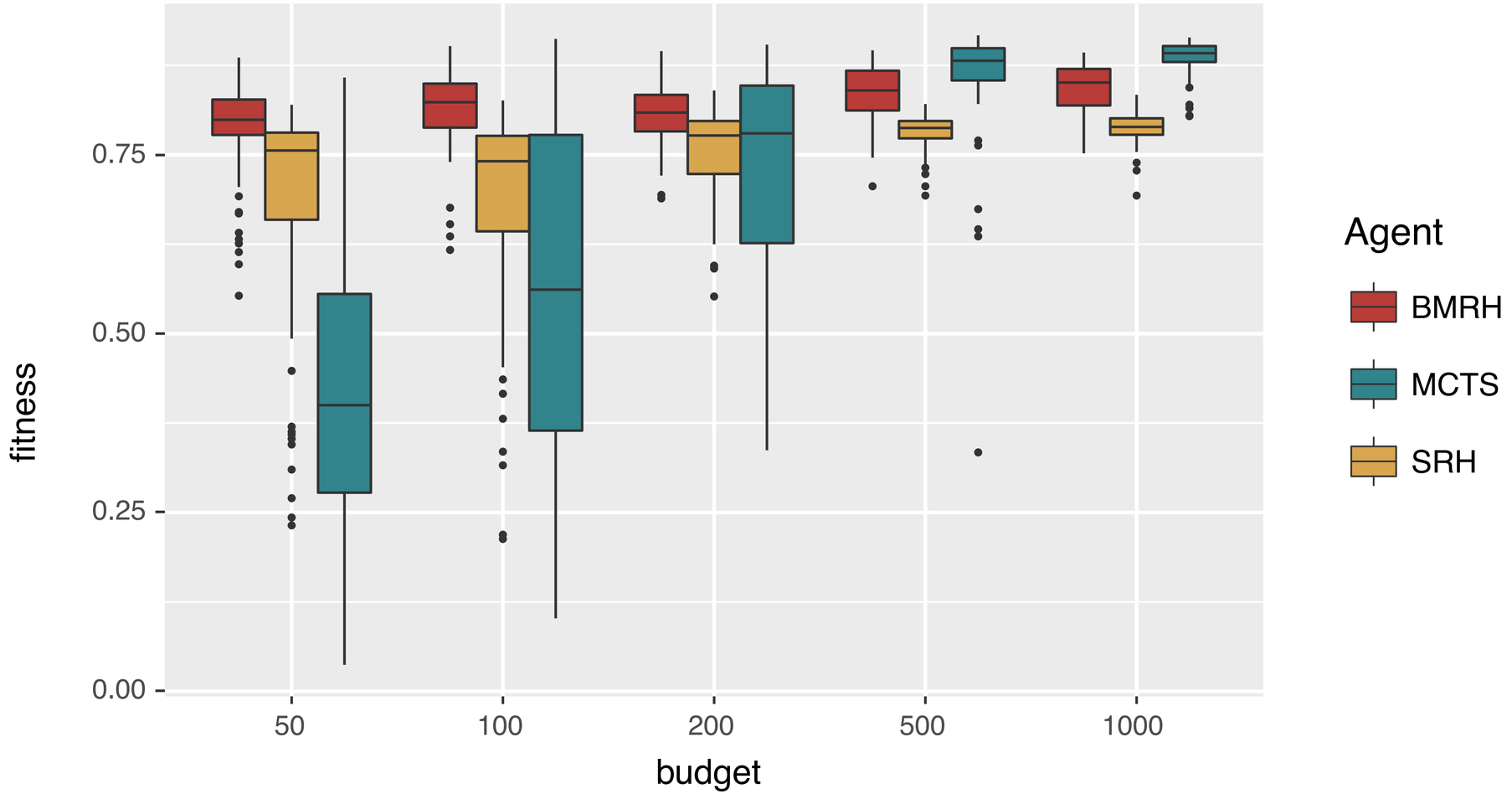
Hyper-parameter Tuning Landscape



Hyper-Parameter Landscape

Agent vs OSLA

100 runs of the NTBEA tuning algorithm per Agent per tuning budget



Round Robin Tournament

Best agents tunings, from previous experiments, against each other.

P1 vs P2	P1 win rate	P2 win rate	SM
MCTS* vs BMRH*	52.3%	47.5%	0.2%
SRH* vs BMRH*	40.2%	59.5%	0.3%
SRH* vs MCTS*	39.8%	59.3%	1.6%

Future Work

seeing ℜ potential

Game Tuning

ℜ defines a space of Splendor-like games, we can search that space using optimisation algorithms to:

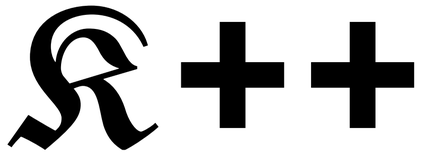
- tune games for specific AI players (max win rate)
- tune players for specific game parameters (max win rate)
- search player styles (max style novelty)

Procedural Content Generation

Deck/Card/Nobles generation!

Do you recognise any pattern?

Modify the actions set



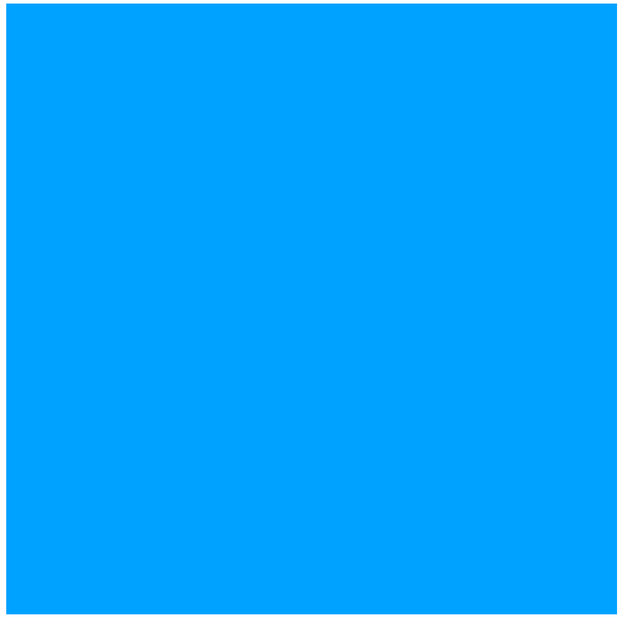
The engine is modular on the playable actions which means it is very easy to add new mechanics given the same base game state.

Rinascimento AI Competition

Hyper-parameter-tuning-based competition

Thanks

@ivanbravi



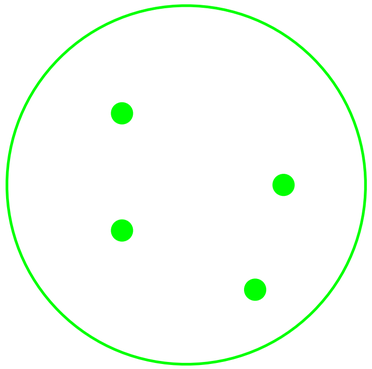
slides

Thanks

@ivanbravi

Population vs Population

Agent A



Agent B