Economics and Computations – Project Presentation

Leduc Poker Solver

11th July 2020

Luca Carminati
Teresa Costa Cañones
Robert Stefano Chinga

Project authors

Luca Carminati
Laurea Magistrale
Computer Science and Engineering

Teresa Costa Cañones
Laurea Magistrale
Telecommunication engineering

Robert Stefano Chinga
Laurea Magistrale
Computer Science and Engineering

POLITECNICO MILANO 1863

OUTLINE:

OUTLINE:

1 ABSTRACTION

2 NASH EQUILIBRIUM APPROXIMATION

3 STRATEGY REFINEMENT

POLITECNICO MILANO 1863

Abstraction

Introduction

Problem : the original game tree may be very large and it makes the computation of a Nash Equilibrium infeasible.

Solution : Use an Abstraction to reduce the original game tree into a new one of a reasonable size without losing much information from the original structure.

Then find a Nash Equilibrium for the new game tree and remap the result into the original.

Our approach to find the abstraction :

- **Game tree analysis** : We analysed the structure of the game tree (Leduc Poker), trying to find some similarities between each sub tree and features regarding the information sets.
- **Research** : We read papers to see different abstractions and different approaches.
- **Discussion** : We discussed different ways to abstract the original game tree and we chose the abstraction.
- **Implementation and evaluation** : After having chosen the abstraction, we implemented it and we analysed the result.

Abstraction

Game tree analysis

Regarding the results about the game tree analysis:

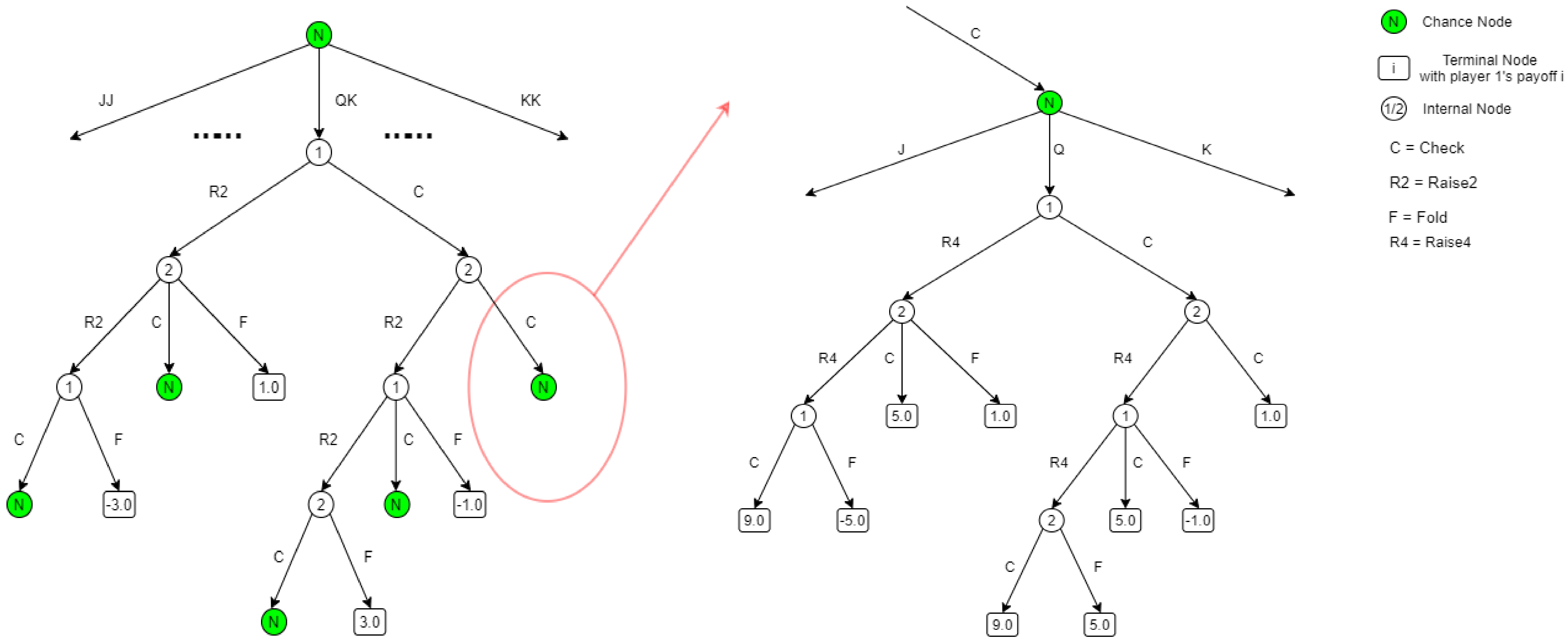
The first and the second round have a similar structure. The only differences are the following:

- In the first round **some path lead to a chance node**. These chance nodes represent the transition between the first and the second round in which the dealer shows the common card.
- In the second round a player can raise 4 chips while in the first round only 2.

The next slide shows the structure of the first round by considering the hand QK (left side) and the structure of the second round by considering the same subtree after the following sequence of actions : Player1 - Check and Player2 – Check (right side).

Abstraction

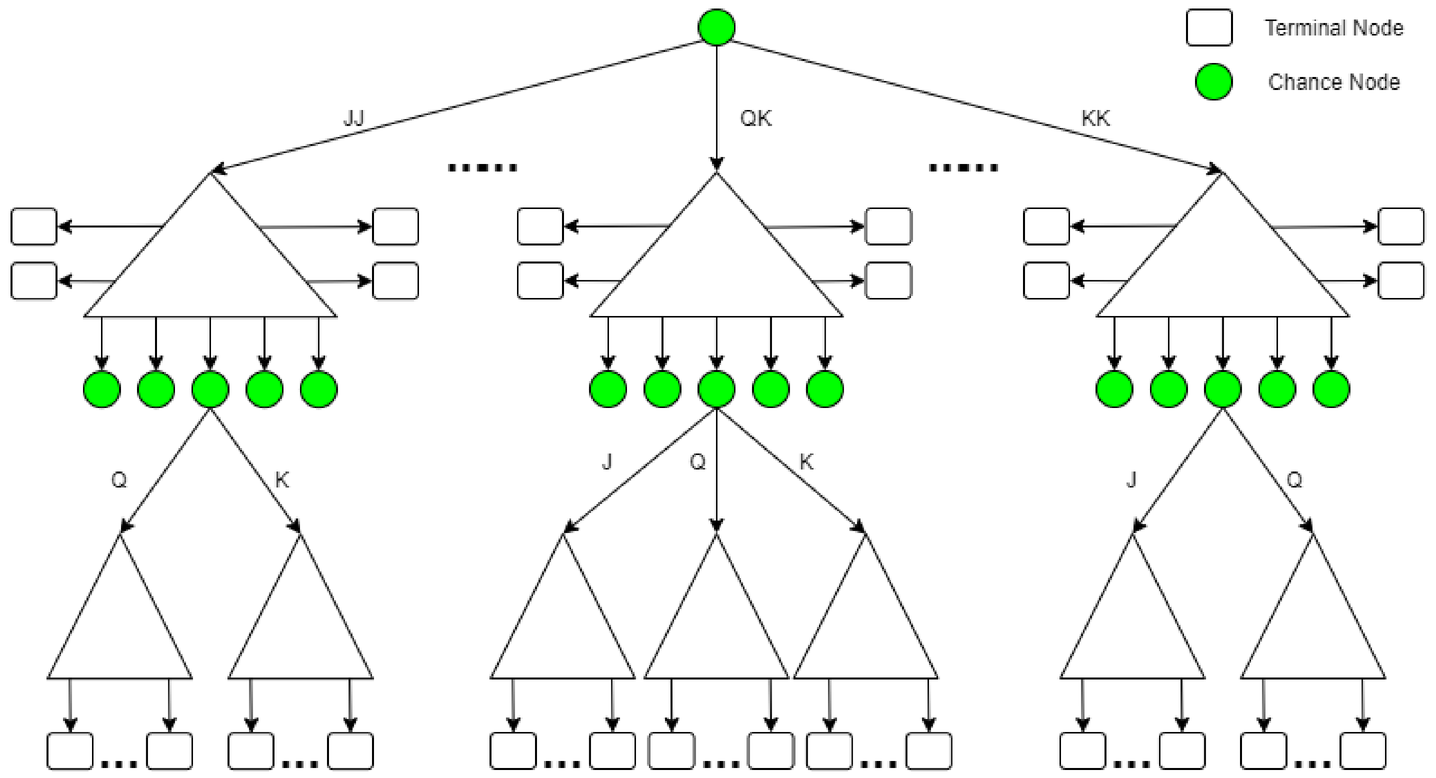
Game tree analysis



Abstraction

Game tree analysis

So it is possible to imagine the original game tree :
Each subtree has a similar structure. For instance, note the hand JJ and the hand KK.



Generally, each subtree differs from one to another for the payoffs and for the internal chance nodes.
We need to pay attention when we merge two subtrees!

Abstraction

Game tree analysis – Information Sets

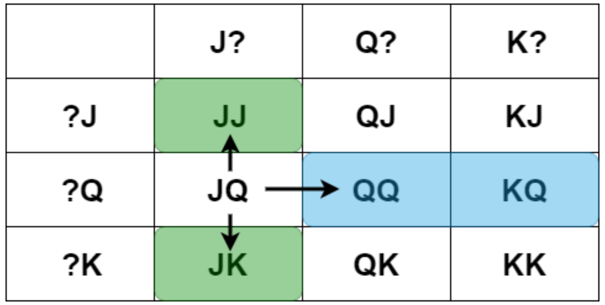
What about information sets?

Each subtree can be associated **to two information sets**: one of player 1 and the other of player 2. This happens because **each player cannot see the private card of the adversary**.

This association can be represented by means of a matrix. Each row indicates an information set of player 2 while each column an information set of player 1. Each entry of the matrix represents a hand and the information set associated to it.

	J?	Q?	K?
?J	JJ	QJ	KJ
?Q	JQ	QQ	KQ
?K	JK	QK	KK

By using this matrix we can see how the subtrees are connected to each other. For instance, the subtree JQ is connected to the subtrees JJ and JK due to player 1 (column J?) but it is also connected to the subtrees QQ and KQ due to player 2 (row ?Q).



	J?	Q?	K?
?J	JJ	QJ	KJ
?Q	JQ	QQ	KQ
?K	JK	QK	KK

Abstraction

Research + Discussion

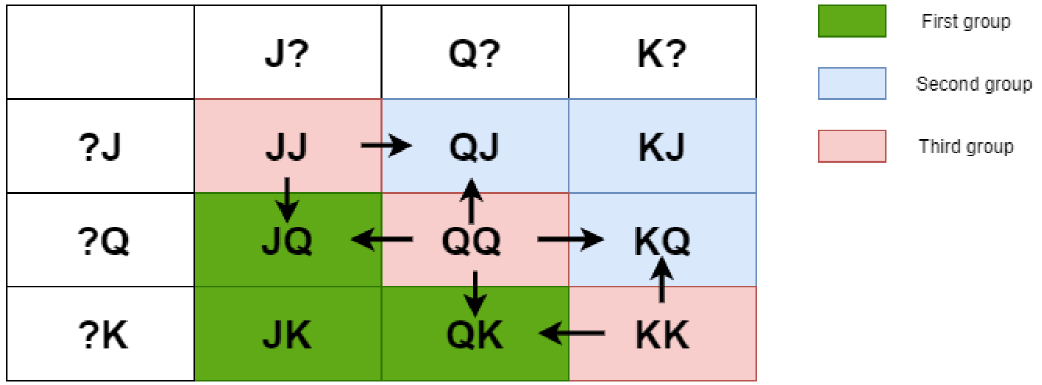
We came up with two ideas:

FIRST IDEA: Use k-means to group the subtrees with similar payoffs. Then create a representative tree for each group. In the end, create the new game tree using only the representative trees.

We found that a good number for k (so for the number of groups) was 3 : one group represents the situation in which the player 1 starts with a better card w.r.t. the player 2, another group represents the vice versa and the last one the situation in which both players have the same card.

Main problems:

1. Increasing the number of cards in Leduc Poker makes this abstraction coarser (25 subtrees in Leduc 5 vs 3 subtrees in the abstraction).
2. Loss of the meaning of the information sets :



Each group is connected to the other for both player1 and player2.

POLITECNICO MILANO 1863

Abstraction

Card binning

SECOND IDEA: Card Binning

Basic idea: **Group the cards by their strength and consider each group as a new card.**

The procedure is the following :

1. Compute the strength of each card by using the expected value of the subtrees in which the card is played by player 1 (for instance, if we want to compute the strength of J in leduc3, we need to consider the subtrees JJ,JQ,JK).
2. Group cards with similar strength (the number of groups is chosen by us).
3. Now each group is considered as a new card and we substitute it in the game.
4. Group the subtrees of the original game tree that have the same name after the substitution.

The idea has its origin by reading the following paper Sam Ganzdriend and Tuomas Sandholm – «Potential-aware Imperfect-Recall Abstraction».

Note: Our approach is much simpler than the one discussed in the paper and it is adapted to Leduc Poker.

Abstraction

Card binning example

Now let's consider an example with Leduc3:

3 cards : J-Q-K

S(i) = strength of card i

We can compute the strength of each card and state the following:
S(J) < S(Q) < S(K)

$\longrightarrow$

create 2 groups:
{J,Q} = J
{K} = K
(Note that we rename the group using a card inside it for simplicity)

$\longrightarrow$

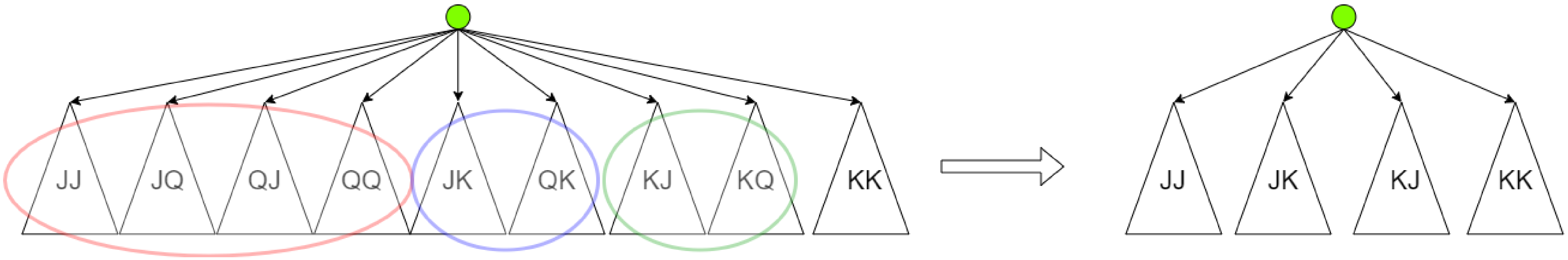
The game change as follows:

JJ	QJ	KJ
JQ	QQ	KQ
JK	QK	KK

$\Rightarrow$

JJ	JJ	KJ
JJ	JJ	KJ
JK	JK	KK

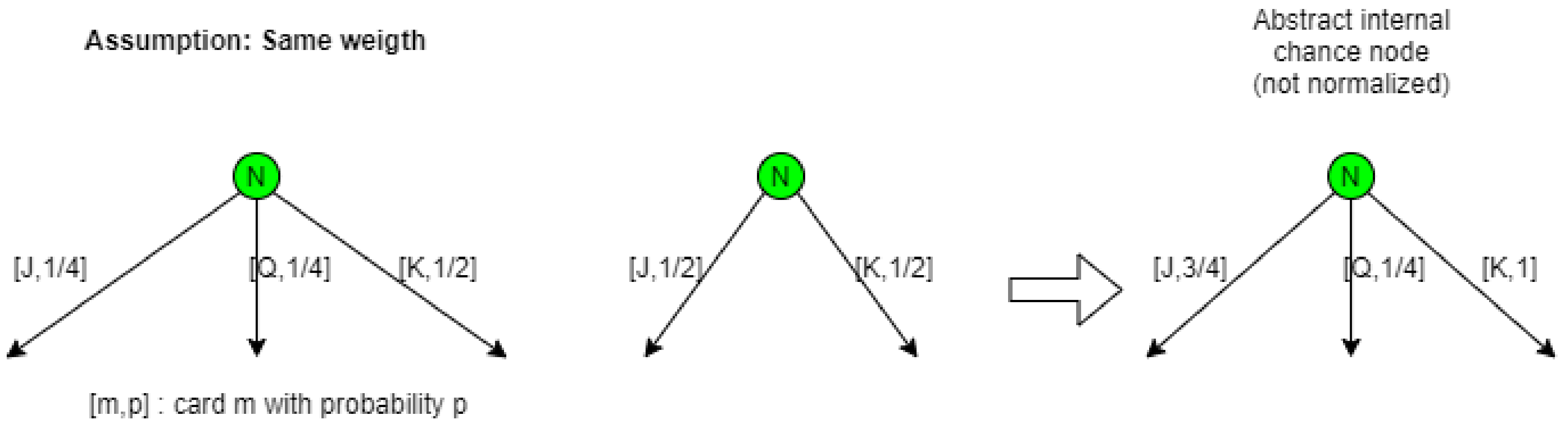
What about the game tree?



Abstraction

How to merge subtrees

- As said in the game tree analysis, we need to be careful when we merge two or more subtrees to obtain a new subtree :
  - **With respect to the payoffs** :The payoffs of the new subtree will be the average of the payoffs of the subtrees belonging to the same group.
  - **With respect to the internal chance nodes:** The new internal chance node will have as set of actions the union between the actions of the internal chance nodes belonging to the original subtrees, and the probability of each action of the abstract chance node will be the sum of the probabilities for such action from the original chance nodes (each probability weigthed by the probability to be in that specific subtree) and then normalized.



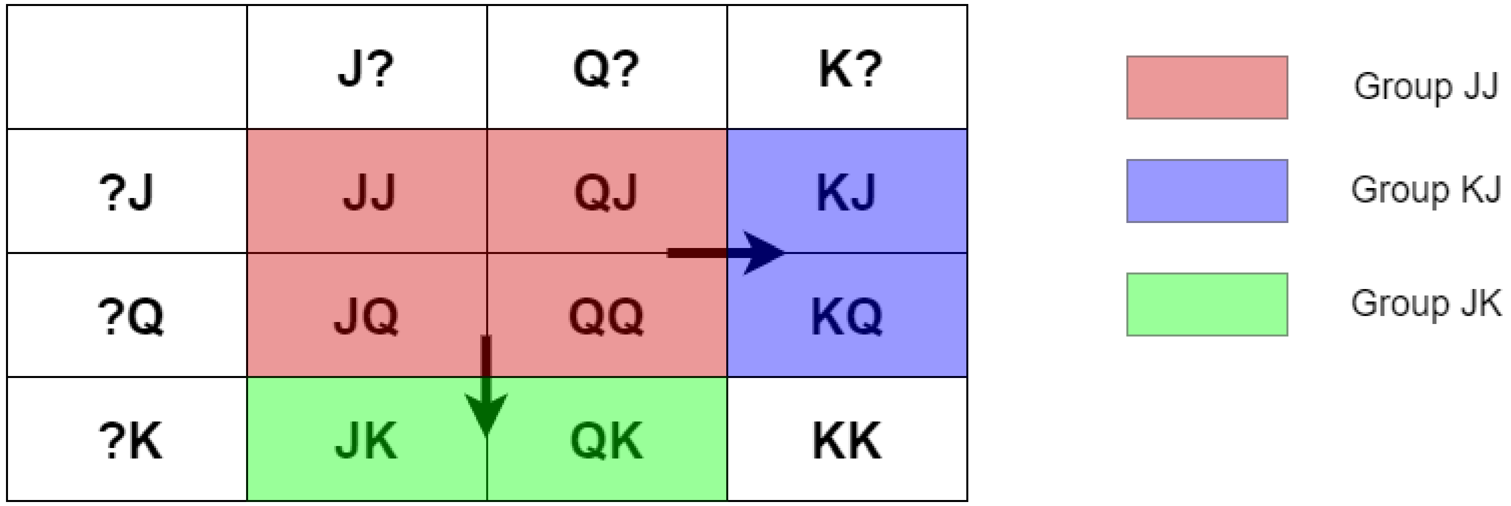
Note: successively cards will also be binned at this level

POLITECNICO MILANO 1863

Abstraction

Final consideration

The card binning abstractions allowed us to solve the problems of the first idea.
In particular, the abstraction maintains the meaning of the information sets.



	J?	Q?	K?
?J	JJ	QJ	KJ
?Q	JQ	QQ	KQ
?K	JK	QK	KK

🟥	Group JJ
🟦	Group KJ
🟩	Group JK

The abstract subtree JJ is connected to the abstract subtree KJ (due to player 2) but also it is connected to the abstract subtree JK (due to player 1) .

OUTLINE:

1 ABSTRACTION

2 NASH EQUILIBRIUM APPROXIMATION

3 STRATEGY REFINEMENT

POLITECNICO
MILANO 1863

NASH EQUILIBRIUM APPROXIMATION

Introduction

GOAL:

Find a Nash Equilibrium in 2-player zero-sum game taking into account that such game tree is big.

Find the blueprint strategy (solve the abstracted game).

PROBLEM:

It is not feasible to do linear programing of a big game tree.

IDEA:

Perform techniques related to the minimization of **regret.**

Use the **Counter Factual Regret minimization** technique (**CFR**) to find which of the different strategies is more suitable for our game.

NASH EQUILIBRIUM APPROXIMATION

Many options are possible:

- *CFR*
- *CFR+*
- *CFR+ averaged*
- *CFR-BR*
- *Discounted CFR*

Each option need to be compared and find the one which has fewer time to converge and the more optimal strategy.

There are more options than the ones introduced.

NASH EQUILIBRIUM APPROXIMATION

Comparison between CFRs

	CFR+	CFR+ averaged	CFR-BR	Discounted CFR
Differences with CFR	• Converges faster than CFR • High compression ratio can be attained	• Converges faster than CFR+	• Don't need to abstract the game because needs less memory	• Converges faster than CFR+

Decision:
CFR+ averaged because converges faster and have better performance.
Implemented accordingly to reference paper: http://arxiv.org/pdf/1407.5042v1
Comparisons:
In order to see which are the differences in a more visual manner it is used http://jeskola.net/cfr/demo/

OUTLINE:

Strategy Refinement

PROBLEM:

Strategies produced by playing the abstracted game are OK-ish in the general case, but they lack effectiveness in the cases of hands made of cards which have been abstracted together.

This because different cards were made undistinguishable to simplify the game (**card binning**)

SO

Exploitability of Final Strategy can be further improved.

IDEA:

perform a **REFINEMENT** of the final strategy **directly on the original game.**

HOWEVER

Computational issues that pushed us towards abstracting the game are still present and must be solved.

We decided to solve them by considering only portions of the subgame at a time, while adopting strategy found in abstract game in the remaining parts of the game.

Strategy Refinement

Introduction

Many options are possible, we opted to implement some of them

- *CFR Refinement*
- *Bias Refinement*
- *Self Generative Refinement*
- *InGame Refinement*

Level-by-level approaches

Depth limited tree approach

Each of these Refinement methods has been tested with respect to **exploitability improvement** and **time of execution.**

REFERENCE PAPER: *N.Brown*- Depth-Limited Solving for Imperfect-Information Games

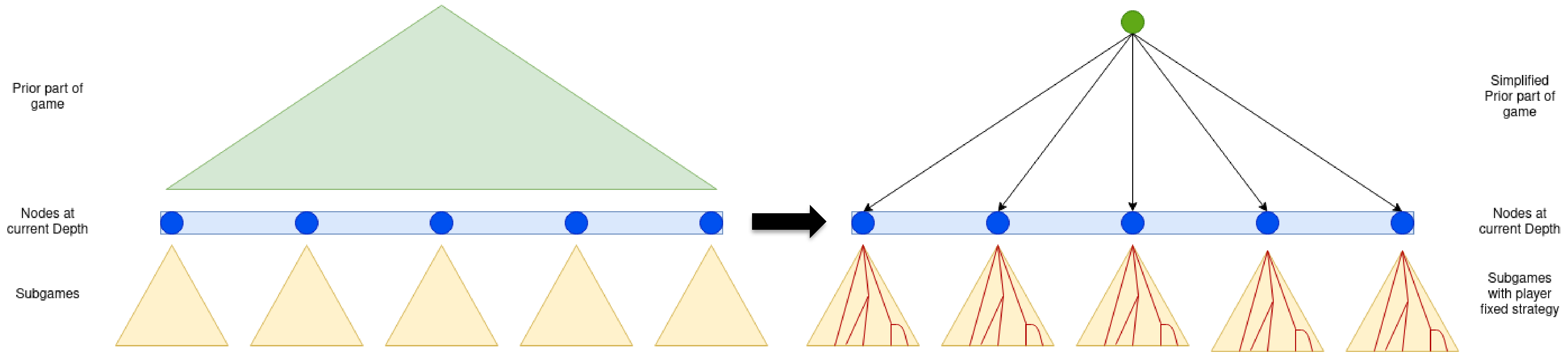
NOTE: To find the best response to a strategy of a player, we modified CFR algorithm to keep the strategy of the player under examination fixed, while changing the strategy of the adversary. Running this algorithm for a very short number of iterations converges to a BR strategy.
This is not the most efficient implementation possible but required little effort to be coded (since most part was already implemented in CFR) and performances are good enough for our setting.

Strategy Refinement

CFR Refinement

CFR Refinement

```
For each level of the complete game
    For each player
        • Substitute the nodes above the current depth with a single root Chance Node.
        • Reach probabilities are those specified by the Blueprint.
        • Fix the strategy of current player in nodes below current depth.
        • Solve this simplified game using CFR+.
        • Use strategy of player found at nodes at current depth in final strategy.
```



Strategy Refinement

Bias Refinement

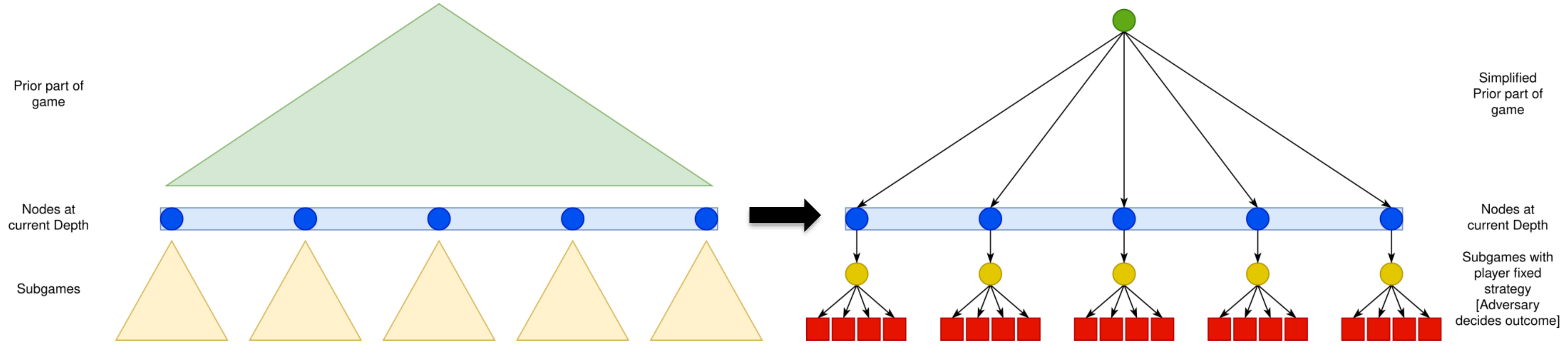
Bias Refinement

For each level of the complete game:

For each player:

- *Substitute the nodes above the current depth with a single root Chance Node.*
- *Fix the strategy of current player in nodes below current depth.*
- *Fix the strategy of adversary to 4 possibilities: Blueprint, Blueprint but probability of Check/Raise/Fold multiplied by 10 at each Decision Node.*
- *Solve this simplified game using CFR+.*
- *Use strategy of player found at nodes at current depth in final strategy.*

All strategies are fixed -> directly compute payoff

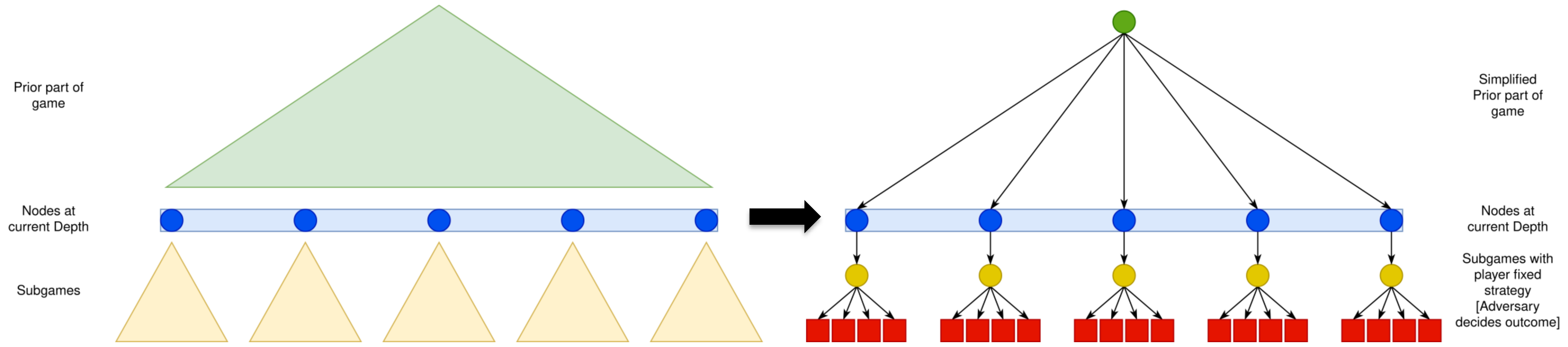

Prior part of game
Nodes at current Depth
Subgames
Simplified Prior part of game
Nodes at current Depth
Subgames with player fixed strategy [Adversary decides outcome]

Strategy Refinement

Self Generative Refinement

Self Generative Refinement

```
For each level of the complete game:
    For each player:
        Do the same as Bias Refinement
        BUT
        Compute final payoff by computing adversary best response to each of the player strategy, iterating
        this procedure until a fixed number of payoffs is reached.
```


Prior part of game
Nodes at current Depth
Subgames
Simplified Prior part of game
Nodes at current Depth
Subgames with player fixed strategy [Adversary decides outcome]

POLITECNICO MILANO 1863

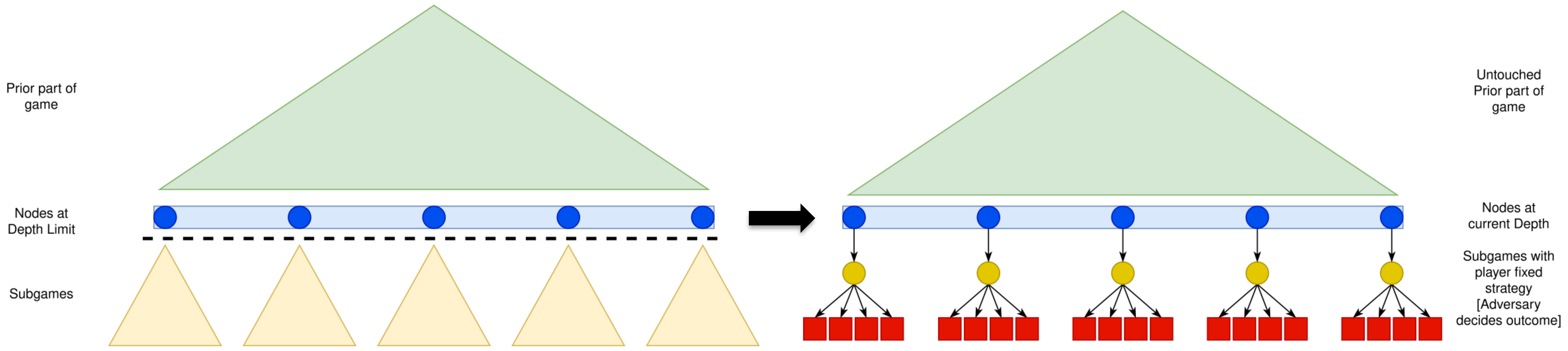
Strategy Refinement

In Game Refinement

In Game Refinement

One-Shot refinement: No level-by level approach

- *Simplify lower subgames by generating possible payoffs in the same way as in Bias Approach.*
- *Refine strategies by using CFR+ on whole remaining tree (= Prior Game + Nodes at Depth limit).*
- *Use found strategy as final strategy in the upper part of the tree, keep blueprint in lower parts.*


Prior part of game
Nodes at Depth Limit
Subgames
Untouched Prior part of game
Nodes at current Depth
Subgames with player fixed strategy [Adversary decides outcome]

Strategy Refinement

Conclusions

From the tests we performed on solving Leduc5 games we acknowledged that:

- Level by level techniques:
  - Suffer from **Badly scaling Performances** due to the level-by-level approach that must repeatedly apply CFR+ for each depth level.

    Note that this is true even if in a more simplified game we can use less iterations to reach a good enough Nash Eq.
  - They are **Unsafe** due to the fact that adversary may react to a strategy taken by the current player in current depth nodes, by changing his strategy in upper levels. However those have already been considered in previous iterations and so the adversary cannot change its strategy anymore.

  This causes an **overall Exploitability increase** with respect to the use of a blueprint strategy computed on a fine-grained abstraction.

SO we opted for <u>In Game Strategy Refinement</u> since:

- **Better Performances** are guaranteed by the one-shot application of CFR+ to a single tree.
- **Safeness** is **theoretically guaranteed** [see referenced paper] by biasing some of the computed payoffs.

Conclusions

Final Remarks

SO OUR CHOICES ARE:

- Card Binning abstraction
- CFR+
- In-Game Refinement

OUR RESULTS ON TESTS:

	Leduc 5 NO ABS	Leduc 5 4 card ABS	Leduc 5 3 card ABS	Leduc 5 2card ABS	Leduc 9 5 card ABS	Leduc 13 5 card ABS
Time Abstraction	-	0.059 s	0.065 s	0.064 s	0.277 s	1.036 s
Time Blueprint	137.18 s	58.19 s	24.70 s	6.90 s	122.46 s	171.95 s
Time Refinement	-	82.76 s	82.44 s	81.02 s	426.30 s	1405.19 s
Expected Value Blueprint	-	-0.0445	-0.0205	-0.0103	-0.0262	-0.0595
Exploitability Blueprint	-	0.246	0.452	0.741	0.224	0.213
Expected Value Final	-0.113	-0.128	-0.147	-0.0624	-0.0874	-0.0838
Exploitability Final	0.000241	0.168	0.288	0.357	0.0790	0.102

Thanks for the attention!

POLITECNICO
MILANO 1863