

Poker Articles 2011

Table of Contents

1 Introduction	4
1.1 Goal of This Document	4
2 Poker Trees	5
2.1 Game Tree	5
2.2 Player Tree	5
2.3 Observer Tree	5
2.4 Chance Tree	5
2.5 Action Tree	5
2.6 Which Areas of a Game Trees are Influenced by a Move?	5
3 Best Response	8
3.1 Description of the Best Response Algorithm	8
3.2 BR for Private Cards	8
4 Equilibrium	9
4.1 What is Exploitation?	9
4.2 Significance of the Sign of Deviation from the Equilibrium	13
4.3 Equilibrium Strategy for Private Cards	13
4.4 Obvious Moves	13
4.5 Best Move against an Equilibrium	17
4.6 Finding Equilibrium by a Linear Program	19
4.6.1 Lemma 1	23
5 Convex Functions	25
5.1 Variables are Dependent	25
6 Bot Development	26
6.1 Important Factors in Bot Development	26
6.1.1 Complex Data Structures	26
6.1.2 Visualization	26
6.1.3 Test Data	26
6.1.4 Reference Data	26
6.2 Tricks and Troubleshooting	26
6.2.1 Strategy for Single Poket for an BR Bot	26

6.2.2 Generated Deals	26
6.2.3 Replay	27
7 <i>Appendix</i>	28
7.1 Typographic Conventions	28
7.2 Glossary	28
7.3 Figures	29
7.4 Tables	29
7.5 Bibliography	29

1 Introduction

1.1 Goal of This Document

This document is catalog of ideas about poker. Some of the topics are:

- various observations worth remembering
- answers to questions about poker theory
- algorithms to solve poker-related problems
- notes about situations from a practical bot development

This document systematizes these ideas and serves both as a workspace for development of new algorithms and as a reference.

2 Poker Trees

2.1 Game Tree

This is a full game tree containing all chance and action nodes.

2.2 Player Tree

This is a view of the game from a perspective of one player. The player does not see the private cards of his opponents.

2.3 Observer Tree

This is a view of the game from a perspective of an external observer. The observer does not see private cards of the players.

2.4 Chance Tree

This tree contains only nodes where the dealer acts. It can be obtained from a game tree by removing all player actions and uniting the remaining clusters.

You can also imagine that the players have only the right to call once in the game definition, and these actions are removed from the tree. So, this is a game tree for a trivial game where the players do not make decisions.

For a two-player game this tree contains complete information about the chance factor of the game and also the complete information to compute a showdown. Actually we can define the chance parameters of the game (which cards can be dealt and with which probability) and the game rules concerning the showdown by a chance tree containing absolute chance probabilities and showdown results in terminal nodes. Chance probabilities have to be stored separately for the case when one player folds.

For a multiplayer game we have to save showdown results for all possible combination of folders where at least 2 players remain active. For example, for a 3 player game we need showdowns for the cases when noone folded or when one of the players folded, 4 in total.

2.5 Action Tree

This is a tree containing only player actions. Chance nodes are removed.

2.6 Which Areas of a Game Trees are Influenced by a Move?

Let a player play a mixed strategy. In the nodes where he acts he chooses probability of his actions. Now, let's consider a move in one particular node in his player tree. What nodes in the game tree are influenced by this move?

To answer this question, let's take a look at the player in position 1 (P1) playing Kuhn poker. Let's say, he holds a K in the pocket and selects whether to raise or to call when P0 calls (node n). His move obviously influences all nodes in his game tree that have n as an ancestor (see Figure 1).

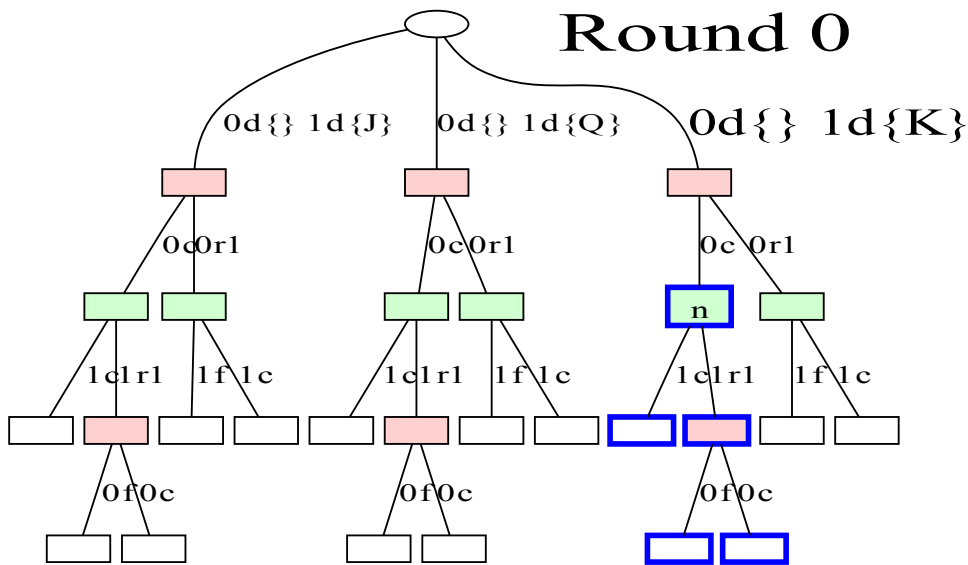


Figure 1: Move of $P1$ in node n .

Now let's have a look at the game tree. The move of P1 is made when he holds K, therefore only such nodes will be influenced. And his move is done after P0 called. In other words, the move influences the nodes with the action path starting with $/0d\{ \}/1d\{K\}/0c$. Notice, that the cards of P0 are unspecified, so it can be a Q or J (K is already dealt).

Figure 2 shows the game tree with influenced nodes shown in blue color. We can see, that the influenced subtrees have the same structure as the subtree in the player tree on Figure 1. Let's call these subtrees clusters. So, the influenced cluster from the player tree is "cloned" in the game tree. The number of clones equals to the number of combinations to deal the cards to the opponents. In our case there are 2 combinations: J and Q.

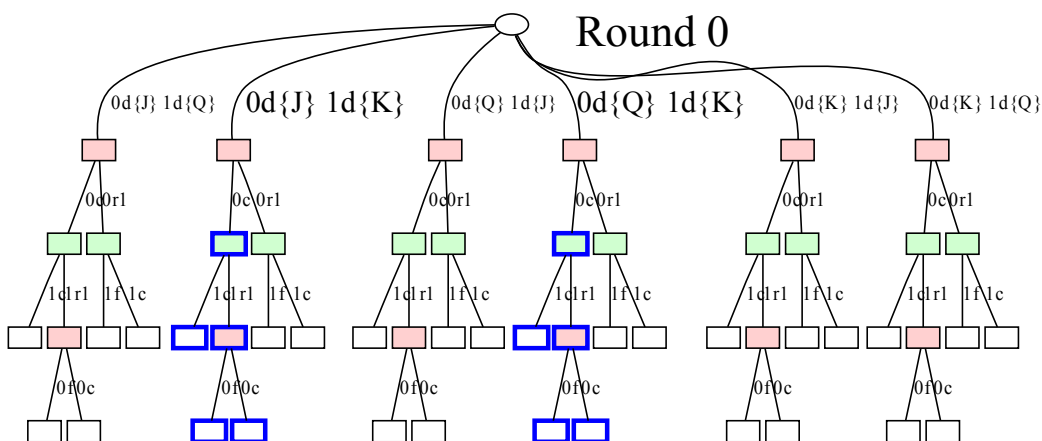


Figure 2: Nodes in the game tree influenced by the move of P1.

When looking at the results of such algorithms as best response, where a player develops a strategy against strategies of the opponent, we can ask: how a move of one player influences the player tree of the opponent? Using the ideas described above, we can answer this question first by projecting player trees of both players into the game tree, and then projecting the game tree back to the player tree of the opponent. This is illustrated on Figure 3.

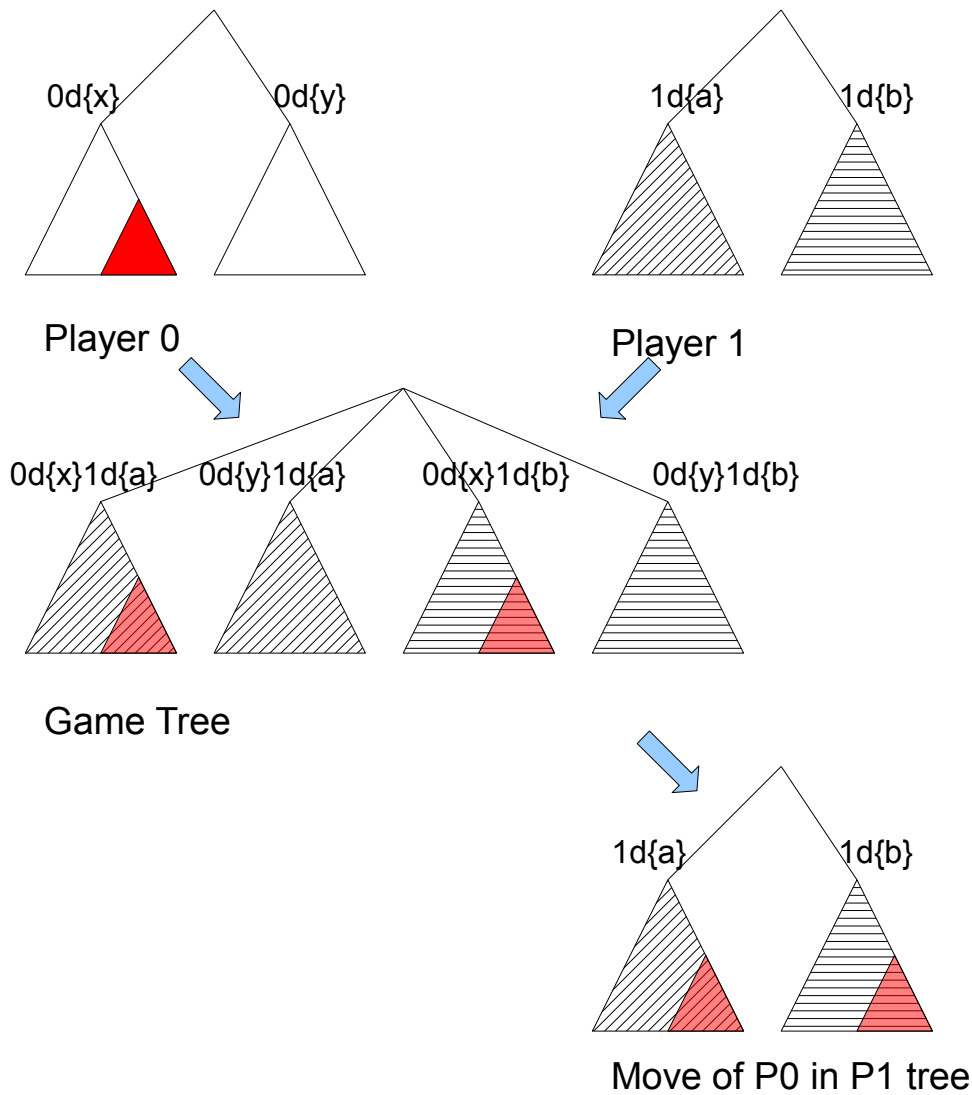


Figure 3: Influence of a move on the player tree of the opponent.

We can see, that the cluster from the player tree of P0 is projected into multiple clusters of the player tree of the opponent. In almost all games a move will be projected to **all** clusters of the tree of the opponent.

A corollary is that any given cluster of the opponent's tree will be influenced by all corresponding clusters of the player's tree.

3 Best Response

3.1 Description of the Best Response Algorithm

Best Response (BR) finds an optimal strategy for a player (hero) against known static strategies of the opponent.

A brief description of the the algorithm is given below:

1. For each opponent's tree: traverse game tree and opponent's tree in parallel. Assign and validate strategic probabilities.
2. Traverse game tree and hero's tree in parallel, calculate values in terminal nodes of the game tree and accumulate them in hero's tree.
3. Traverse hero's tree in postorder marking best response edges and calculating game values.

3.2 BR for Private Cards

BR can be found for each private card combination independently. This is obvious if you look at the picture describing step 3 of BR (Figure 4). The hero make moves in pink nodes, minimizing the values of direct children. Therefore each move influences only the children node. It is clear that branches of the tree corresponding to different private card holdings (J, Q, K) are independent and cannot influence each other.

When there are multiple private deals in different rounds (like in 7-Card Stud), the situation is a little more complex, but is quite similar to the one described above.

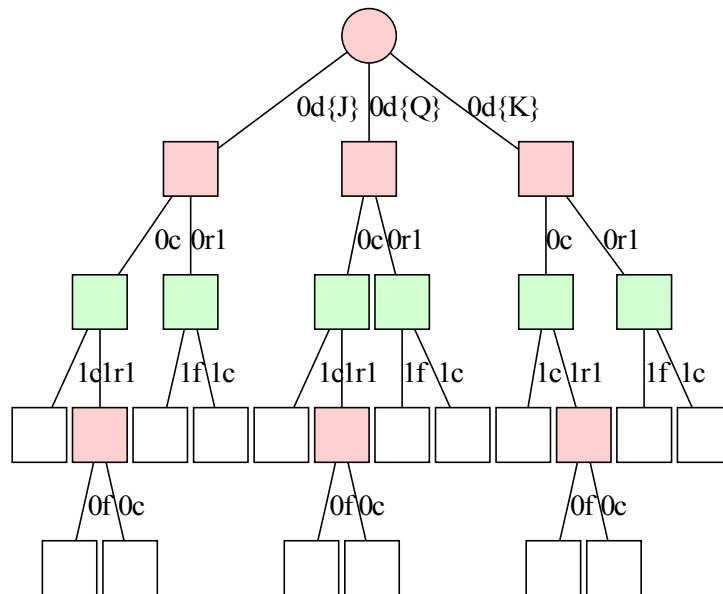


Figure 4: Player tree of the BR hero.

4 Equilibrium

4.1 What is Exploitation?

Here we will explain what happens if a player deviates from the equilibrium strategy. An opponent can use this to increase his game value. This process actually is the exploitation.

For the illustration we will use Kuhn poker. Player in position 1 (P1) will play different static mixed strategies.

Player in position 0 (P0) will try to exploit the strategies of P1 by using best response algorithm.

Figure 5 shows the equilibrium strategy of P1, described in [1].

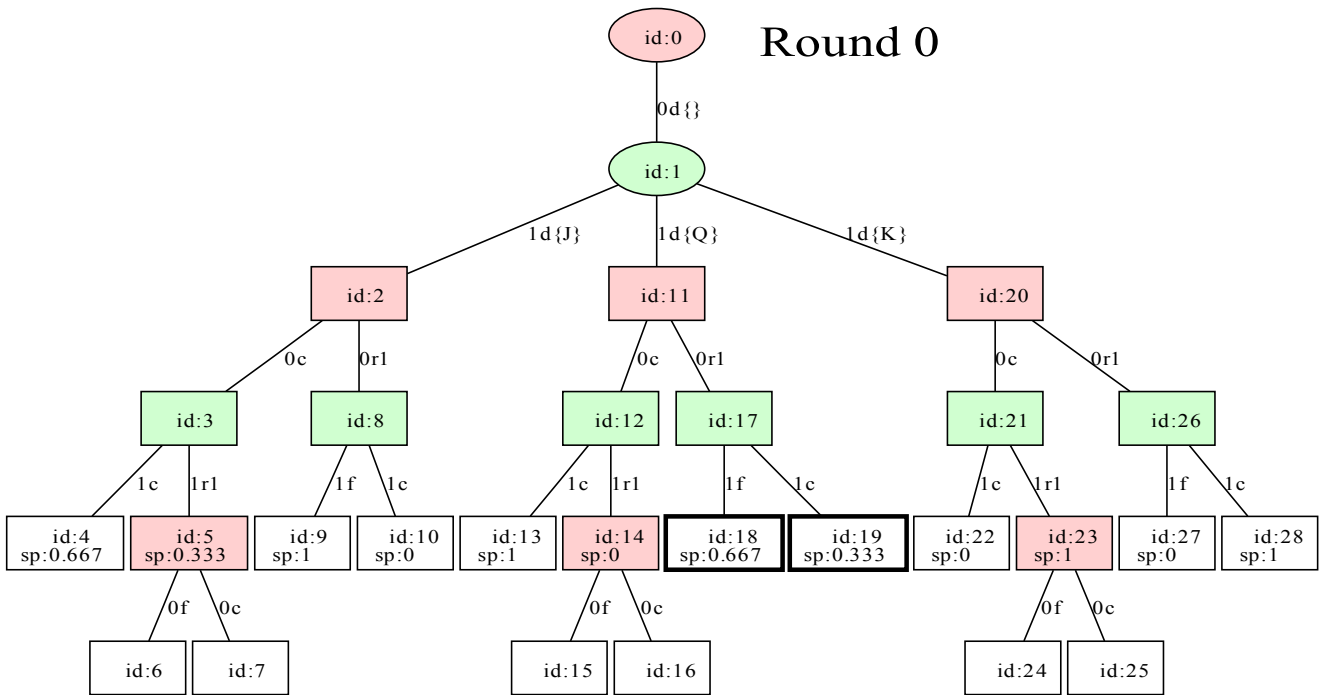


Figure 5: Equilibrium strategy of player 1.

Let's consider one move of P1 that is done in node 17, let's call it m_{17} . In this situation P1 holds Q in the pocket and has to decide if he calls or folds. There is only one equilibrium probability tuple for this move,

described by local probabilities of choosing fold or call: $m_{17} = (\frac{2}{3}, \frac{1}{3}) = (0.667, 0.333)$.

We will first show the result of using the equilibrium probability tuple for this move. Than we will show what happens, if P1 deviates from the equilibrium.

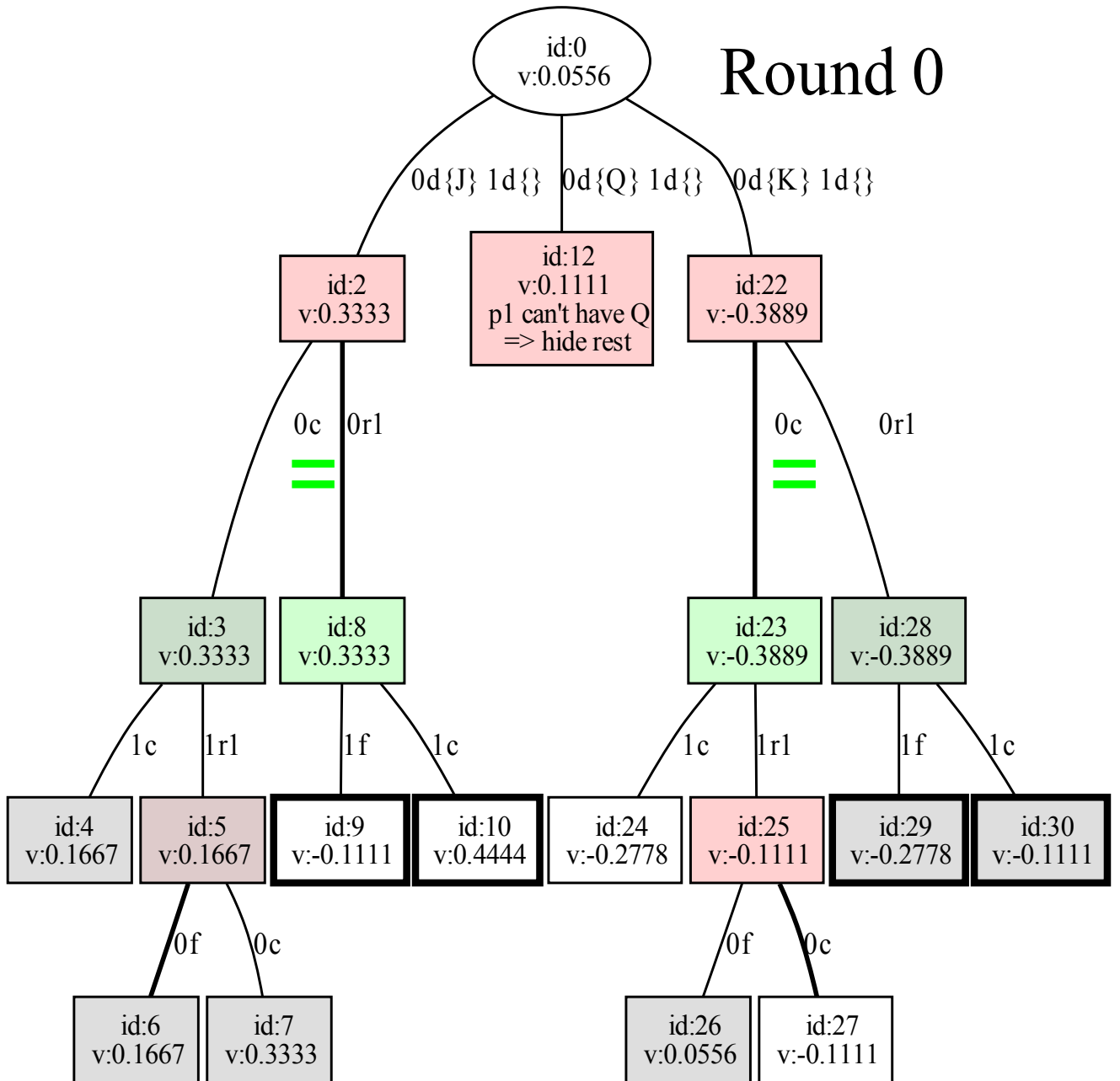


Figure 6: Best response of P0 on equilibrium strategy of P1: (0.667, 0.333).

Figure 6 shows the best response of P0 to the equilibrium strategy of P0. The picture shows game values from the point of view of P1, who is trying to maximize them.

The result of m_{17} directly influences the values in terminal nodes 9, 10, 29 and 30 (drawn with thick lines).

Notice that in nodes 2 and 22 P0 has no best choice, because the values in the child nodes are equal.

Now let's see what happens if P1 will fold more frequently. Let's consider the probability tuple

$$m_{17}^{f+\Delta} = (0.767, 0.233) \quad \text{with the probability of folding increased by 0.1.}$$

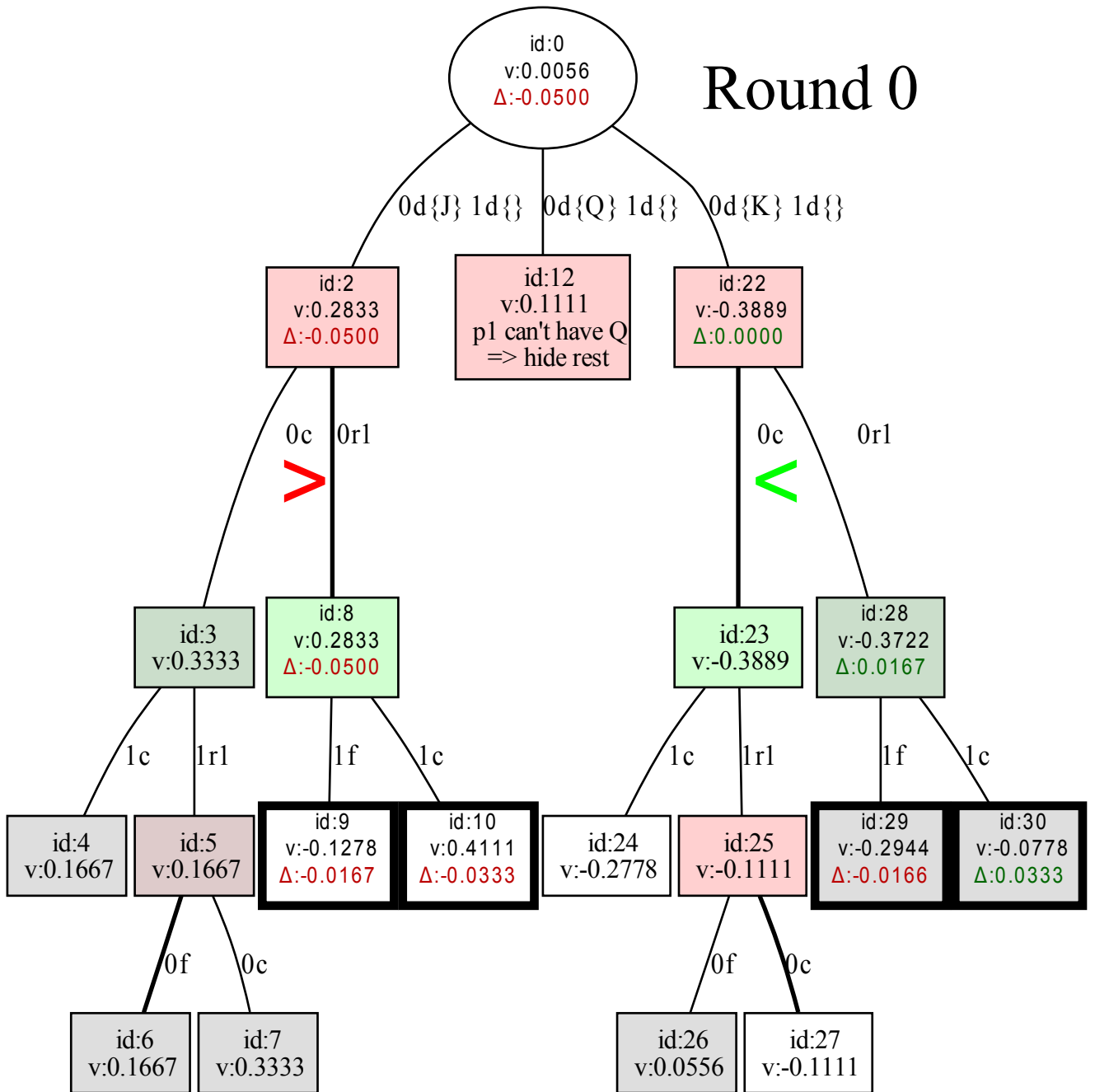


Figure 7: Best response of P0 on strategy of P1 with more folds: (0.767, 0.233).

Figure 7 shows the result of this move. Values in terminal nodes 9, 10, 29, 30 reflect the new probability of P1 having Q in these nodes.

Let's have a look at node 2. P0 is trying to minimize the value. He now has a choice of the best move (raise), and the value of this move is worse for P1 ($\Delta = -0.0500$) compared to the equilibrium. So, P0 can take advantage of the better for him value.

In the node 22 P0 also has a choice of the best move (call). Although the node 28 now has a positive Δ , P1 cannot take advantage of it, because P0 neutralizes this advantage by choosing another node. As the result, the value in node 22 remains the same as for equilibrium.

The overall game value in node 0 has a negative Δ , so P1 wins less than in the equilibrium.

In other words, the following happens:

1. The new move changes the values of some nodes, some deltas are positive, some are negative
2. P0 takes advantage of negative deltas and neutralizes positive deltas

A very similar situations happens if P1 will change the move in the other direction and fold less frequently, for example $m_{17}^{f-\Delta} = (0.567, 0.433)$, as shown on Figure 8.

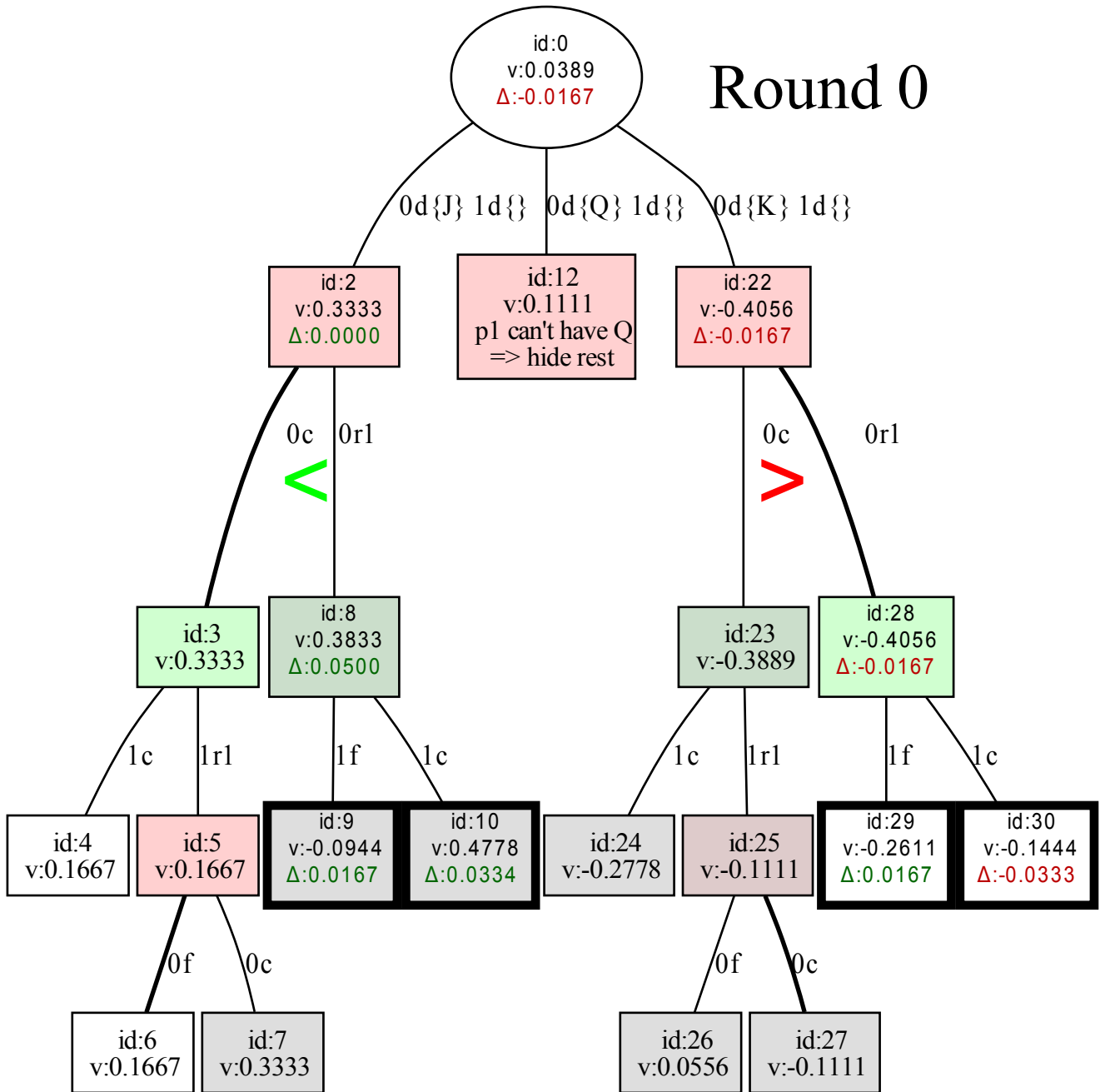


Figure 8: Best response of P0 on strategy of P1 with more calls: (0.567, 0.433).

Now P0 wins in the node 22 and parries in node 2.

To sum up, the exploitation is the ability of a player to neutralize the advantage of the opponent and take advantage of good opportunities for himself.

4.2 Significance of the Sign of Deviation from the Equilibrium

Let's remember a mental experiment described in 9. In this experiment the probability of folding deviated from the equilibrium strategy first by +0.1, then by -0.1, in other words, by the same absolute delta. But the overall game value changed by different deltas.

This happens because the final change in game value comes in each case from two different nodes (8 and 28), as shown in Table 1.

Move	Value in node 8	Value in node 28
$m_{17}^{f+\Delta}$	$\Delta_v = -0.0500$	$\Delta_v = +0.0167$
$m_{17}^{f-\Delta}$	$\Delta_v = +0.0500$	$\Delta_v = -0.0167$




Table 1. Influence of different moves on node values.

As you can see, the same absolute delta of folding probability actually changes the game values by the equal absolute deltas in the each node in our example. But due to the fact that P0 choses every time the node with negative delta, different deltas are propagated up the tree.

This shows the significance of the sign of the probability change in probability tuples. Algorithms that are trying to find the equilibrium by estimating game value as a function of the probability of a move have to take this into account.

4.3 Equilibrium Strategy for Private Cards

In constrast to BR, equilibrium strategy cannot be found for each private card combination of a player separately. A counterexample is Kuhn Poker. As known from ([1]), it has multiple equilibrium strategies for position 0, where moves with different pockets depend on each other. I proved this practically by calculation of BR for different strategies of player 0.

4.4 Obvious Moves

It is obvious that you will never fold the nuts and never call a raise with the weakest holding possible to see a showdown. But what are the weakest holding and nuts from the theoretical point of view. And are there other situations where one move is clearly should be favored without worrying about exploitation (see 4.1)?

Let's use Kuhn poker as an example (2). Figure 9 and Figure 10 show strategy trees of player 0 and 1 with variables that describe the absolute strategic probability of playing a node.

KunhPoker hero tree pos 0

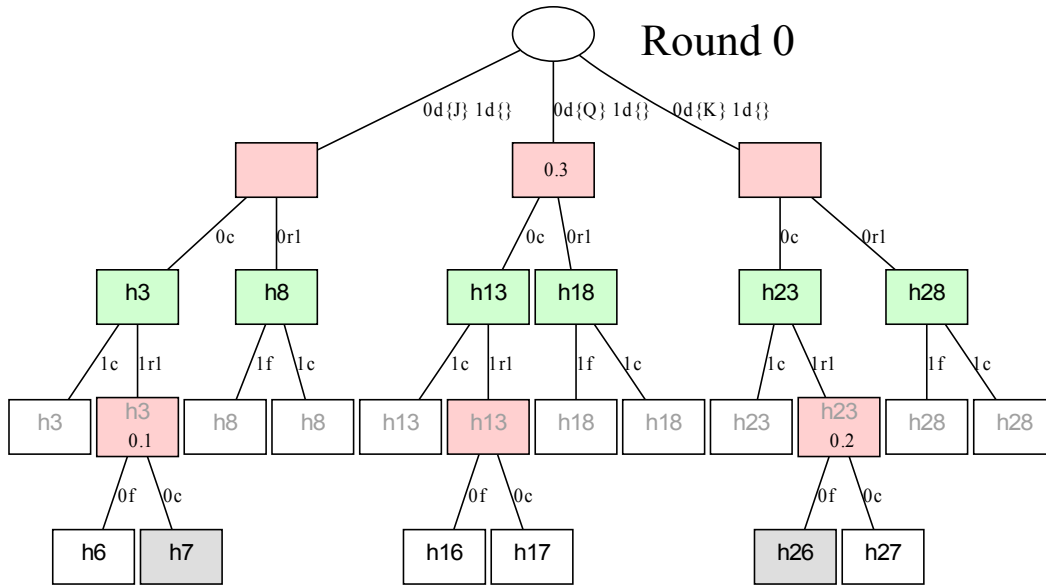


Figure 9: Strategy of player 0.

KunhPoker hero tree pos 1

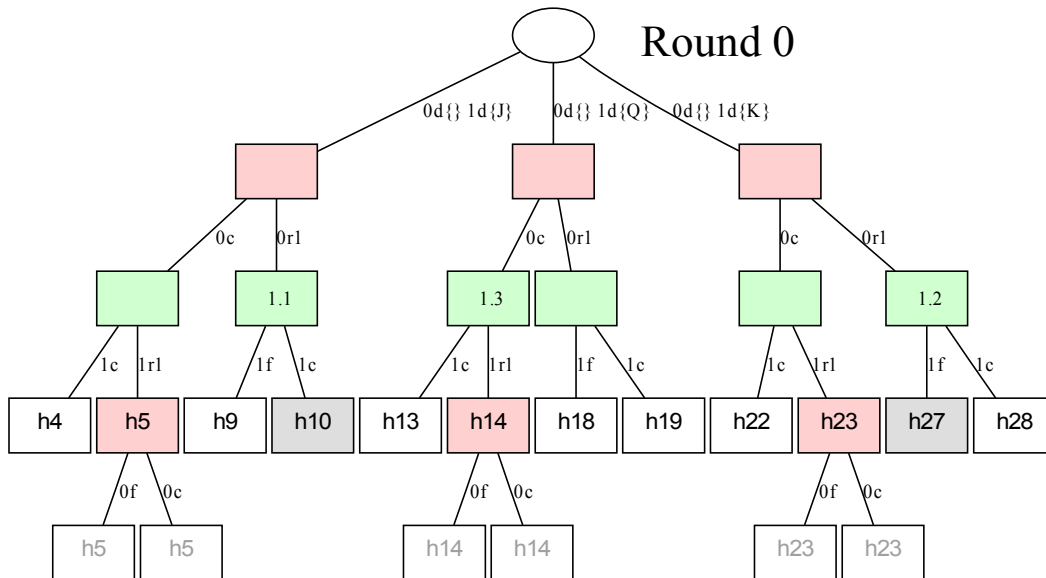


Figure 10: Strategy of player 1.

Below is a qoutation from (2) adjusted for our notation:

Obviously, no player will decide either to bet on a J (0.1: $h_6=1$, $h_7=0$; 1.1: $h_9=1$, $h_{10}=0$) or to pass with a K (0.2: $h_{26}=0$, $h_{27}=1$; 1.2: $h_{27}=0$, $h_{28}=1$) when confronted by a bet.

We numbered nodes in questions using convention *position.number* and assigned values to the strategic probabilities. Nodes with zero probabilities are shown grayed on Figure 9 and Figure 10.

Let's examine one of these moves, for example 0.1. How to prove that this choice cannot be exploited? To do this, let's have a look on the opponent tree playing the BR (Figure 11).

KunhPoker opponent tree for hero pos 0

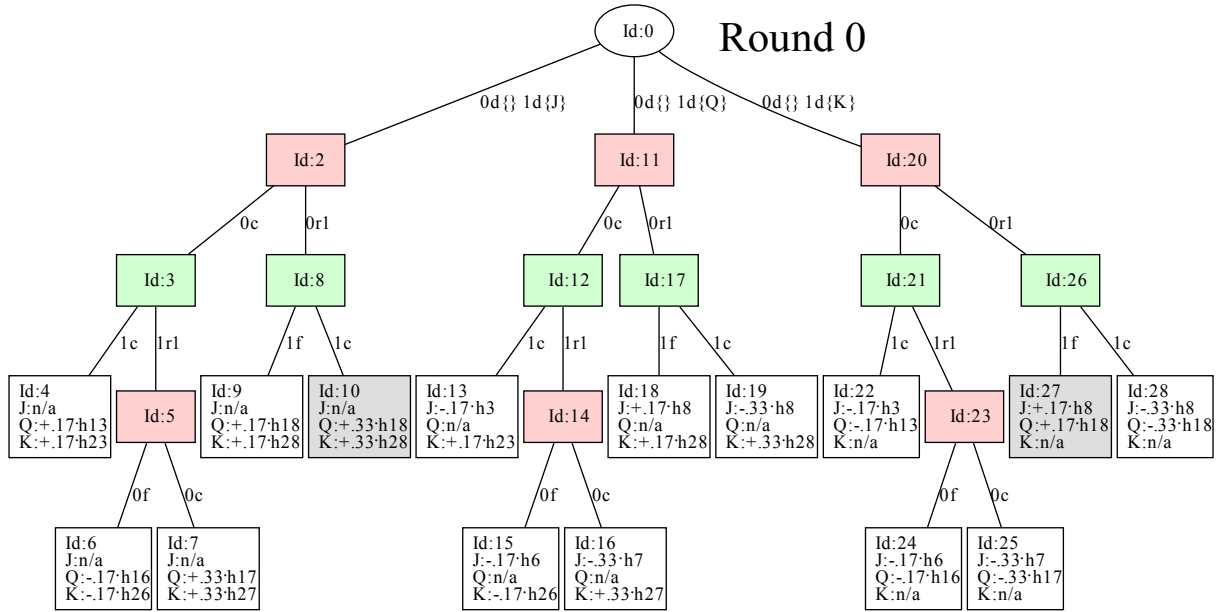


Figure 11: Opponent tree for hero position 0.

KunhPoker opponent tree for hero pos 1

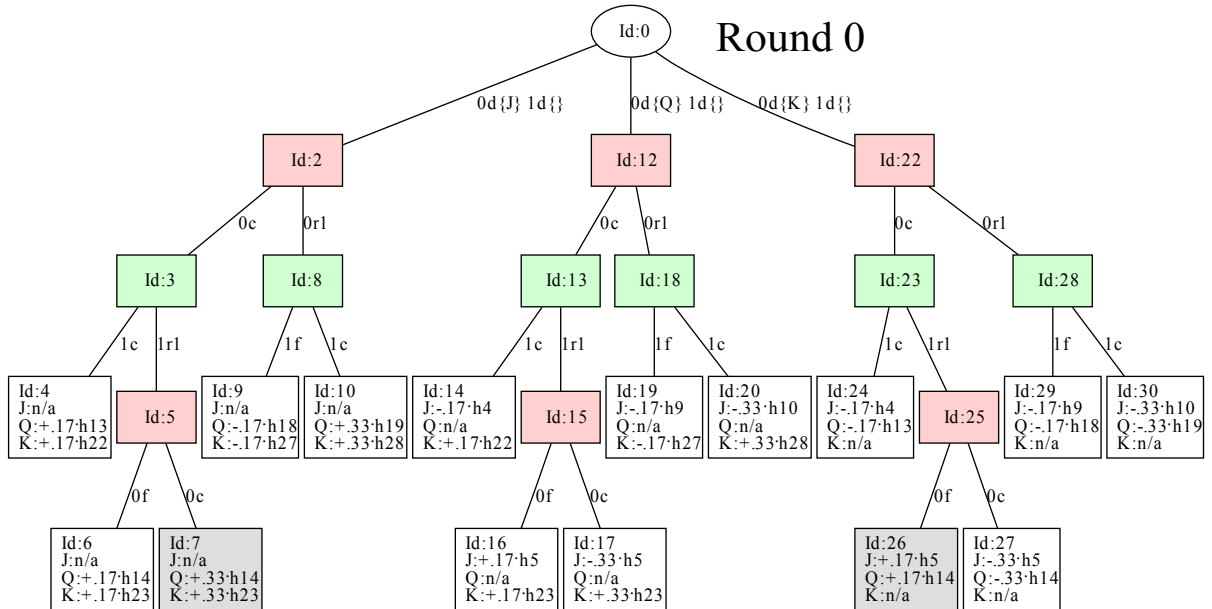


Figure 12: Opponent tree for hero position 1.

In terminal nodes there are possible holdings of player 0 with the corresponding utility function coefficients and strategy variables. We are interested in nodes containing variables h6 and h7. The information is summarized in Table 2:

Tree cluster	$1d\{Q\}$	$1d\{K\}$
Coeff. for h6	-0.17	-0.17
Coeff. for h7	-0.33	-0.33

Table 2. Data for move 1.0.

Observe that h6 has a greater coefficient in both clusters. If player 0 prefers to play h6 instead of h7, he will get a greater game value in **both** node 14 and 23. Regardless of how the opponent defence in nodes 12 and 21, player 0 will get equal or greater overall game value.

We can generalize this rule (see Figure 13):

$$\text{If } \forall \text{ cluster } i: c_{1i} \leq c_{2i} \text{ then } h_2 \text{ should be preferred.} \quad (1)$$

We can make shure that this rule apply for moves 0.2, 1.1 and 1.2.

Actually by preferring a move we eliminate a set of strategies that contain a complementary (h_1) for that move. This strategies are all dominated by a corresponding strategies that contain the preferred move.

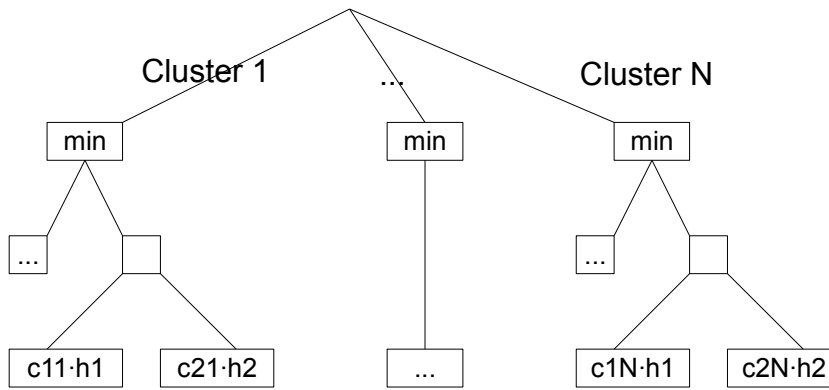


Figure 13: A schematic view of the BR opponent tree.

Are there other situations where we can prefer a particular move? Below is another a citation from (2):

Now that these strategies have been eliminated new dominations appear. First, we notice that if player 0 holds a Q he may as well pass in the first round, deciding to bet in the second if confronted by a bet, as bet originally (0.3: $h13 = 1, h18 = 0$). On either strategy he will lose the same amount if player 1 holds a K; on the other hand, player 1 may bet on J if confronted by a pass but certainly will not iif confronted by a bet. Secondly, player 1 may as well pass a bet, when holding a Q and confronted by a pass (1.3: $h13=1, h14=0$) since player 0 will now answer a bet only when he holds a K.

Indeed, now we can ignore nodes with dead moves in both hero and opponent tree. They are shown grayed on diagrams. Let's look at the move 1.3 where player 1 makes a choice between h13 and h14. Again, we can see that in all clusters on Figure 12 h13 has a greater or equal coefficient and should be preferred.

Notice that in this case the obvious move does not mean a dominating strategy, because the definition of domination requires that a dominant strategy must be better than the dominated strategy **regardless** of what the opponent is doing. Here we made an assumption about the move of the opponent. If we made another assumption that the opponent will not act rational and fold the K to a raise and never raises, then the other strategy will be better (raise with the Q afert a call: $h13 = 0, h14 = 1$).

We will not consider the move 0.3 because we think that it cannot be explained so easily. We think that the explanation of the author is incomplet though correct. First we have to proove why we can ignore the move h16, and than procced as explained in the citation.

Maybe this iterative process of selecting obvious moves can be programmed and applied in equilibrium solvers. Probably it will not significantly reduce the complexity of the problem. For example, as described in (3) applying an iterated elimination of dominated strategies for Rhode Island Hold'em reduced the number of variables from 1,237,238 to 1,181,084 (4.5%) in a losslessly shrunked game and from 91,224,226 to 89,121,538 (3.5%) in the original game.

Maybe there are other situations where a move selection is obvious. This is an open question for future research.

4.5 Best Move against an Equilibrium

After reading 4.1 one can come to a conclusion that if equilibrium is an unexploitable strategy, the hero must not give the opponent a chance to choose a best move in any node where the opponent acts. In other words, the values in child nodes of nodes where the BR-opponent is acting must be equal.

Unfortunatly it is not as simple. Let's take Kuhn poker as an example, player 1 is playing an equilibrium strategy and player 0 is playing an BR (Figure 14).

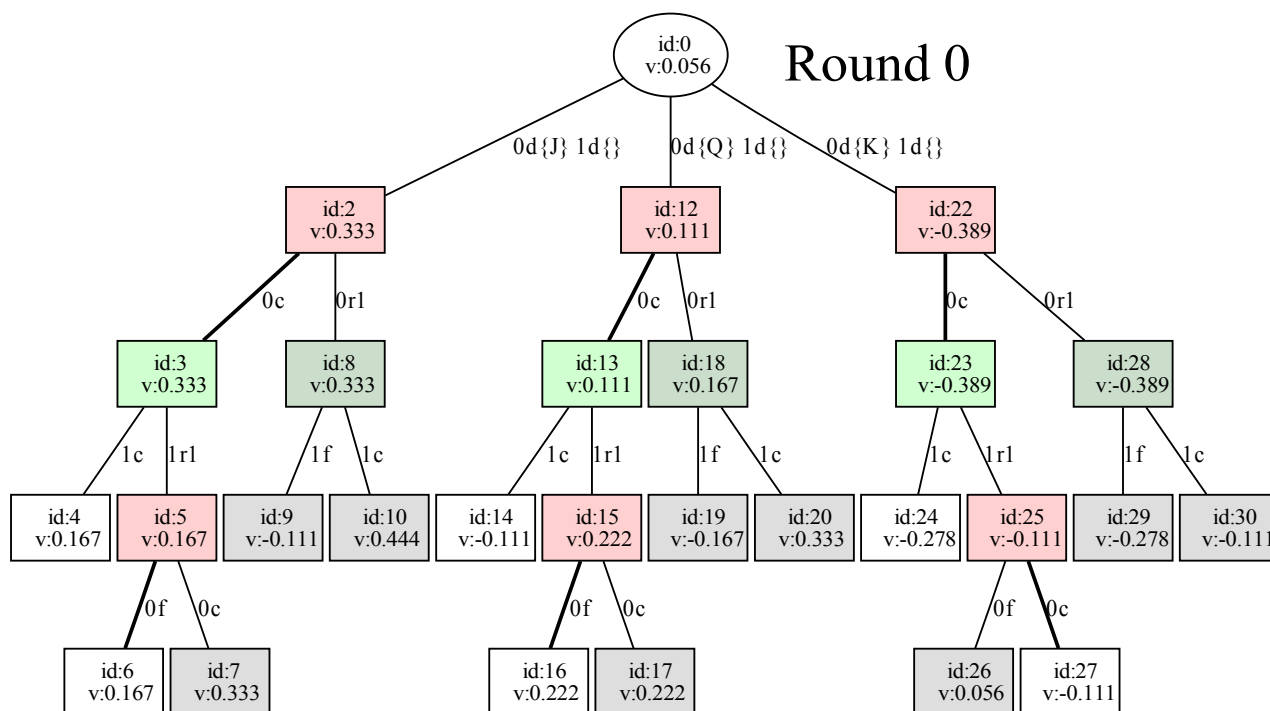


Figure 14: Best response of player 0, values from point of view of player 1.

First, let's look at the node 5. Here player 0 has to decide whether to call or to fold with a J against a raise. The answer is obvious. He must fold, because the J is the weakest holding possible. If he calls, player 1 will always win more regardless of what strategy he is playing.

So, this is one class of nodes with obvious moves (see also 4.4) where the values in children nodes are always unequal. This holds for any strategy, not only for the equilibrium.

Now let's look at node 12. Here an opponent again has a good choice of minimizing the value by playing node 13. Is it possible for player 1 to adjust his strategy so that he will not give player 0 a chance to steal here?

Notice that decisions of player 1 in nodes 13 and 18 are independent. Obviously he either can reduce the value in node 18 or increase the value in node 13 hoping that other nodes will not be worse.

Let's start with node 18. Player 1 holds either J or K (see Figure 15). His strategy with these cards after a raise is obvious: he must fold the J and call with the K. So, it makes no sense to change the strategy in node 18, it is obviously already optimal.

Now let's have a look at node 13. It's clear that the criterion (1) holds for raise with the K. So, the only hope we have is to manipulate the J.

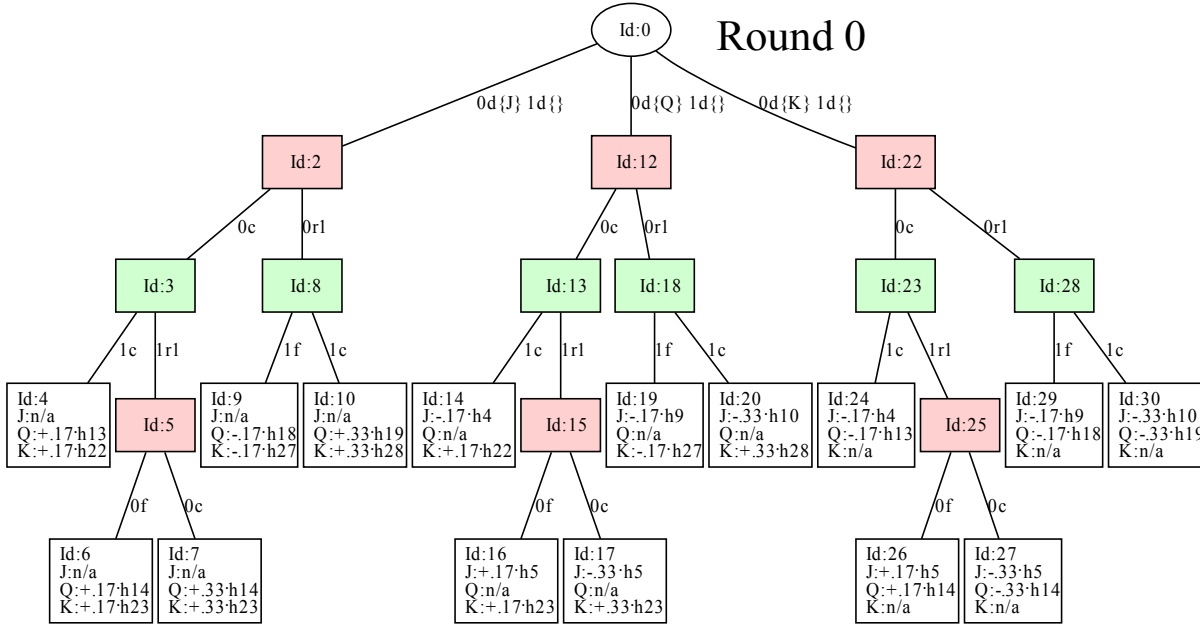


Figure 15: Tree of player 0 with strategy variables of player 1.

We can use a mixture of calls and raises with the J ($h_5 \in [0..1]$). In response, player 0 can either play node 16 or 17. Let's denote the value in node 13 when player 0 plays node 16 as f_{16} and the other value as f_{17} . These are functions of variable h_5 . Then:

$$f_{16} = -\frac{1}{6}(1-h_5) + \frac{1}{6}h_5 + \frac{1}{6} = \frac{1}{3}h_5$$

$$f_{17} = -\frac{1}{6}(1-h_5) - \frac{1}{3}h_5 + \frac{1}{3} = \frac{1}{6}(1-h_5)$$

$$v_{13} = \min(f_{16}, f_{17})$$

Since the values of both functions in node 14 are equal, we can write for the value in node 13:

$$v_{13} = \min(f_{16}, f_{17})$$

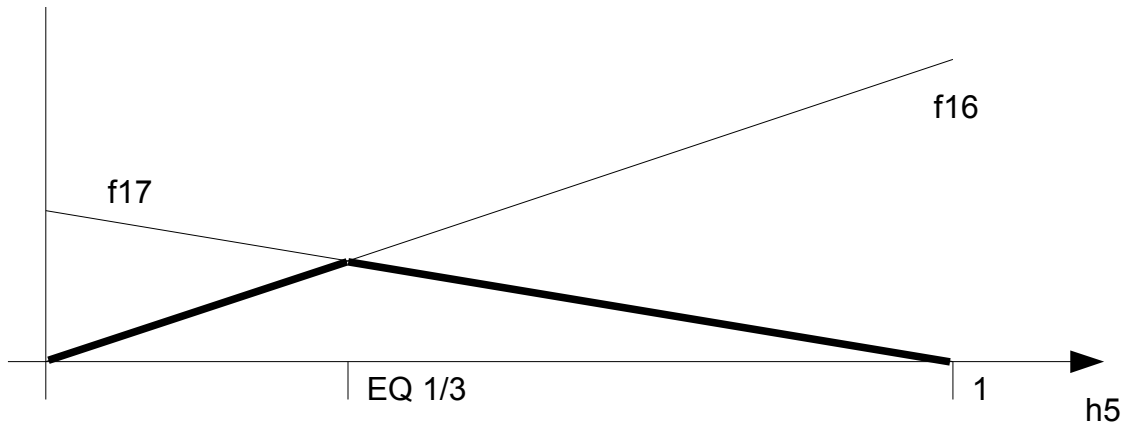


Figure 16: f_{16}, f_{17} and v_{13} (black line).

Figure 16 shows the graphs of these functions. It is clear, that the equilibrium is reached in one point $h_5 = \frac{1}{3}$. All other values are worse. Moreover, there is no hope that we compensate this in the node 23 where player 1 also holds a J. Even when the value there is greater ($h_5 < \frac{1}{3}$), player 0 will counter by playing node 28.

So, the final conclusion is that an equilibrium **cannot** be built by trying to reach the state where the opponent has equal values for all his decisions.

4.6 Finding Equilibrium by a Linear Program

The equilibrium in a poker game can be found by formulating a linear program and solving it for instance with a simplex method. This article describes how to find an equilibrium in a 2-player game. It is an open question, if it can be applied to a multi player game.

We will formulate a linear program that solves the following problem: a player (hero) wants to maximize his game value against a static best response of the opponent. As usual for the best response, the opponent is aware of the strategy of the hero. The hero has to find a strategy with a value greater or equal than all other strategies.

We will describe building the LP step by step using Kuhn Poker ([1], [2]) as an illustration. The hero will be the player in the position 0. The opponent in position 1 will apply BR against the hero.

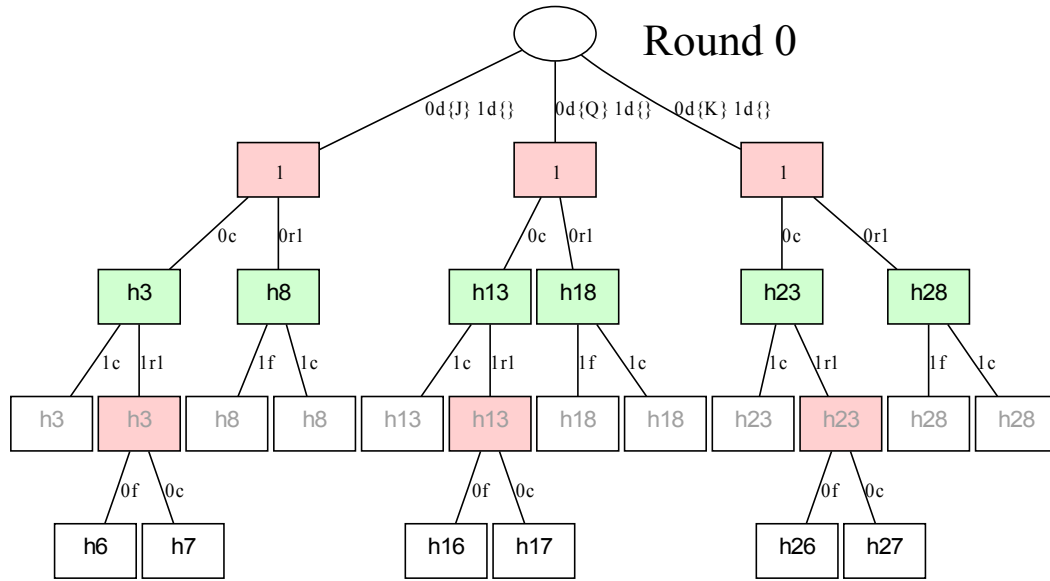


Figure 17: Hero tree.

First, let's have a look at the step 1 in the BR (Figure 17). Here the strategic probabilities of the hero moves are assigned to nodes of the player tree of the hero. Let's denote by h_i the absolute strategic probability of the hero being in node i . Notice that:

1. The probability changes after moves of the hero and remains the same in nodes representing moves of the other players or the dealer. Therefore we introduce new variables only in nodes where the hero has acted, and propagate them to other nodes (shown shaded on the picture).
2. For any node i where the hero acts (pink on the picture) the following is true:

$$h_i = \sum h_{child}$$

$$0 \leq h_i \leq \infty$$

3. The probabilities before the hero has made his very first move is 1. Therefore we do not introduce variables for them. For these nodes the following is true:

$$1 = \sum h_{child}$$

This is the first set of constraints and bounds for the LP.

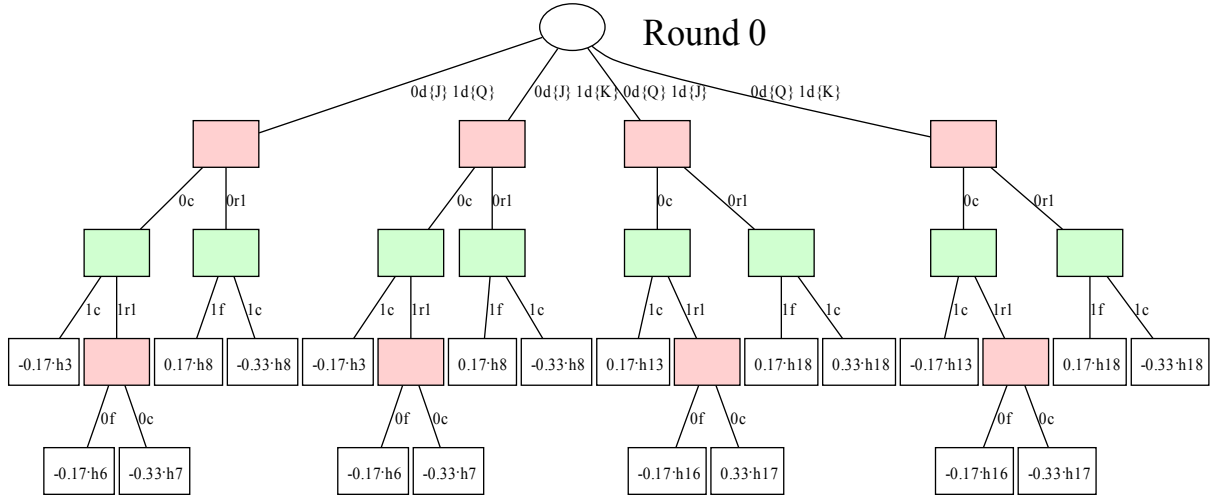


Figure 18: Game tree. The branches where the hero holds K are not shown.

Now, let's consider step 2 of the BR (Figure 18). Here we project the variables from terminal nodes of the hero tree to the terminal nodes of the game tree. When neither player folded, the coefficients for node i can be calculated as following:

$$k_i = s_i * s_{ci} * sh(c_h, c_o, c_{sh})$$

where

s_i : in-pot of each player (equal for both) in the node i

p_{ci} : absolute chance probability of node i

sh : showdown function

c_h : cards of the hero

c_o : cards of the opponent

c_s : shared cards

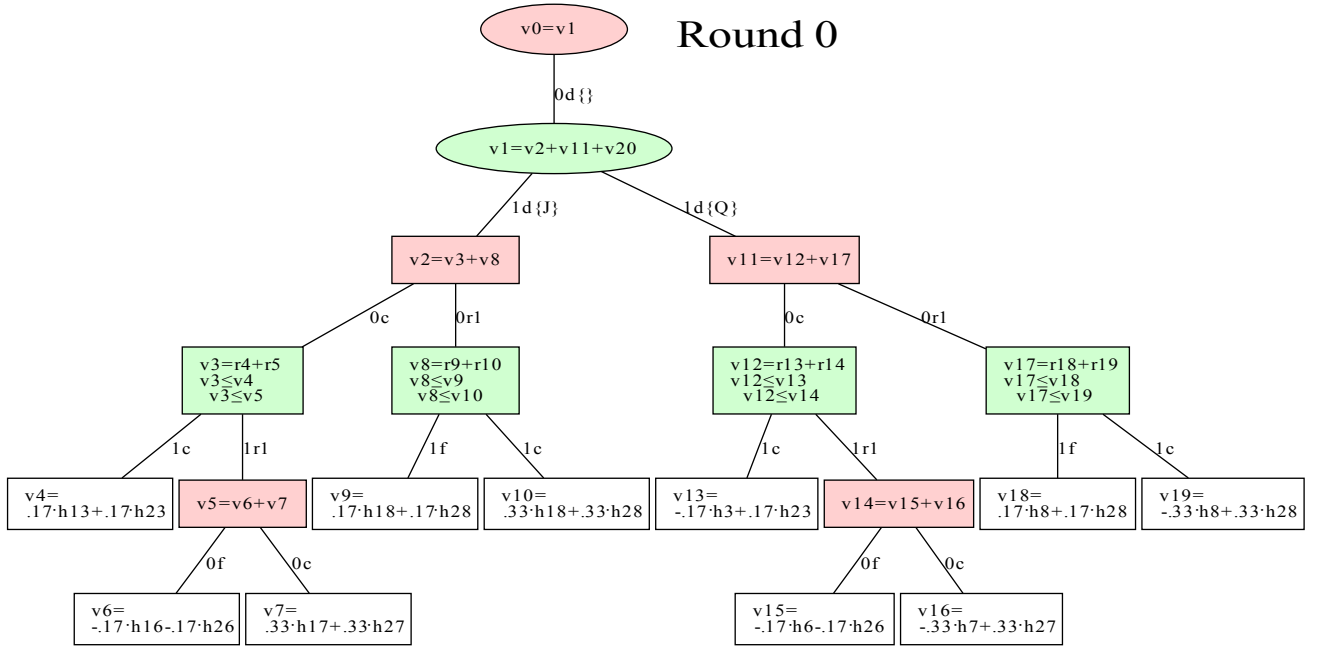


Figure 19: Opponent tree. The branches where the opponent holds K are not shown.

Now let's take a look at the step 3 of BR (Figure 19).

The showdown function returns 1, 0, or -1 depending on the result (the hero wins, ties or loses).

In case one player folded:

$$k_i = s_i * p_{ci}$$

where

s_i : in-pot of the folder (hero: negative, opponent: positive) in node i

p_{ci} : absolute chance probability of node i

For terminal nodes we introduce:

$$v_i = \sum k_j h_j \quad \text{for each nodes } n_j \text{ of the hero tree that are projected to the node of the opponent tree.}$$

$$-\infty \leq v_i \leq \infty$$

There is a special case for nodes where the opponent folds before the hero makes a single move. An example is the small blind in a Holdem game dropping his bad pocket cards without calling the big blind. In this case the constraint has the following form:

$$v_i = \sum k_j$$

$$-\infty \leq v_i \leq \infty$$

Although the showdown result is the same for each k_j , they can differ because of different chance probabilities of playing this node depending on private cards of the hero.

For nodes where the opponent acts we have:

$$v_i = \min v_j$$

$$-\infty \leq v_i \leq \infty$$

for all j from set of child nodes

We cannot use this for the LP. We need another way of representing \min so that it can be written as a linear function and a set of constraints. To do this, let's remember that the opponent can also use a mixed strategy. Then:

$$v_i = \min(o_j * v_j)$$

$$\sum o_j = 1$$

$$-\infty \leq v_i \leq \infty \tag{2}$$

where o_j is a conditional probability of the opponent selecting node j

Now replace the variables:

$$r_j = o_j * v_j$$

$$-\infty \leq r_j \leq \infty$$

It can be proven that we can now rewrite (2) the following way (see Lemma 1):

$$v_i = \sum r_j$$

$$v_i \leq v_j$$

$$-\infty \leq v_i \leq \infty$$

$$-\infty \leq r_j \leq \infty$$

for all j from set of child nodes

Now we can use this for the linear program.

For other nodes we introduce:

$$v_i = \sum r_j$$

$$-\infty \leq v_i \leq \infty$$

This is the other set of constraints and bounds for the LP problem.

The objective is to maximize the game value for hero:

$$v_0 \rightarrow \max$$

These LP can be solved by the simplex method. Values of h_i give the optimal strategy for the hero, values of the best response can be derived from v_i and r_i .

Note that the number of h_i , v_i , r_i is (roughly) proportional to the number of nodes in the player tree. Number of constraints is also roughly proportional to the number of nodes in the player tree.

4.6.1 Lemma 1

Given:

$$f = \min(x_1, x_2, \dots, x_n) \quad (3)$$

Let's introduce new variables:

$$y_i = x_i z_i$$

$$\sum z_i = 1 \quad (4)$$

Then (3) is equivalent to:

$$f' = y_1 + y_2 + \dots + y_n$$

$$f' \leq x_1$$

$$f' \leq x_2$$

$$\dots$$

$$f' \leq x_n \quad (5)$$

Proof:

Let's take $\forall x_i$. We will show that $f = f'$.

We can assume that $x_1 \leq x_2 \leq \dots \leq x_n$ (can be done by reindexing). Then $f = \min(x_1, x_2, \dots, x_n) = x_1$.

Observe, that from all inequalities in (5) only the first is meaningful, the others are corollaries and can be ignored.

Let's denote:

$$x_2 = x_1 + \Delta_1$$

$$\dots$$

$$x_n = x_1 + \Delta_n$$

$$\Delta_i \geq 0 \quad (6)$$

Then the first inequality in (5) can be written as:

$$z_1 x_1 + (x_2 + \Delta_2) z_2 + \dots + (x_n + \Delta_n) z_n \leq x_1$$

$$z_1 x_1 + x_2 z_2 + \Delta_2 z_2 + \dots + x_n z_n + \Delta_n z_n \leq x_1$$

$$x_1 (z_1 + z_2 + \dots + z_n) + \sum_{j=2}^n \Delta_j z_j \leq x_1$$

Remember from (3) that $\sum z_i = 1$, then:

$$\sum_{j=2}^n \Delta_j z_j \leq 0 \quad (7)$$

Let's split Δ_j in two groups: $\Delta_k > 0$ and $\Delta_m = 0$. Then (7) is equivalent to:

$$\sum_k \Delta_k z_k \leq 0. \text{ This holds only if } z_k = 0.$$

Then only z_m can be > 0 . Observe, that from (6) follows $x_m = x_1$. Then

$$f' = \sum y_i = \sum z_i x_i = \sum z_m x_m = x_1 \sum z_m = x_1 = f.$$

■

5 Convex Functions

5.1 Variables are Dependent

The maximum of a convex function cannot always be found by finding the maximum on the first variable where the others are fixed, then the second, etc. There are cases where this process fails, in both sequential (where each variable is processed once) and iterative (where we can repeatedly process a variable) variants. To see this, look at Figure 20.

$f(x, y) \rightarrow \mathbb{R}$ is a convex function. x, y have constraint as show on the picture. The global maximum in in the point m .

If we are at the starting point $(0, 0)$, then maximizing of $f(x, 0) \rightarrow \max$ will lead us to nowhere, because the function is constant $f(x, 0) = 0$. The same is true for y .

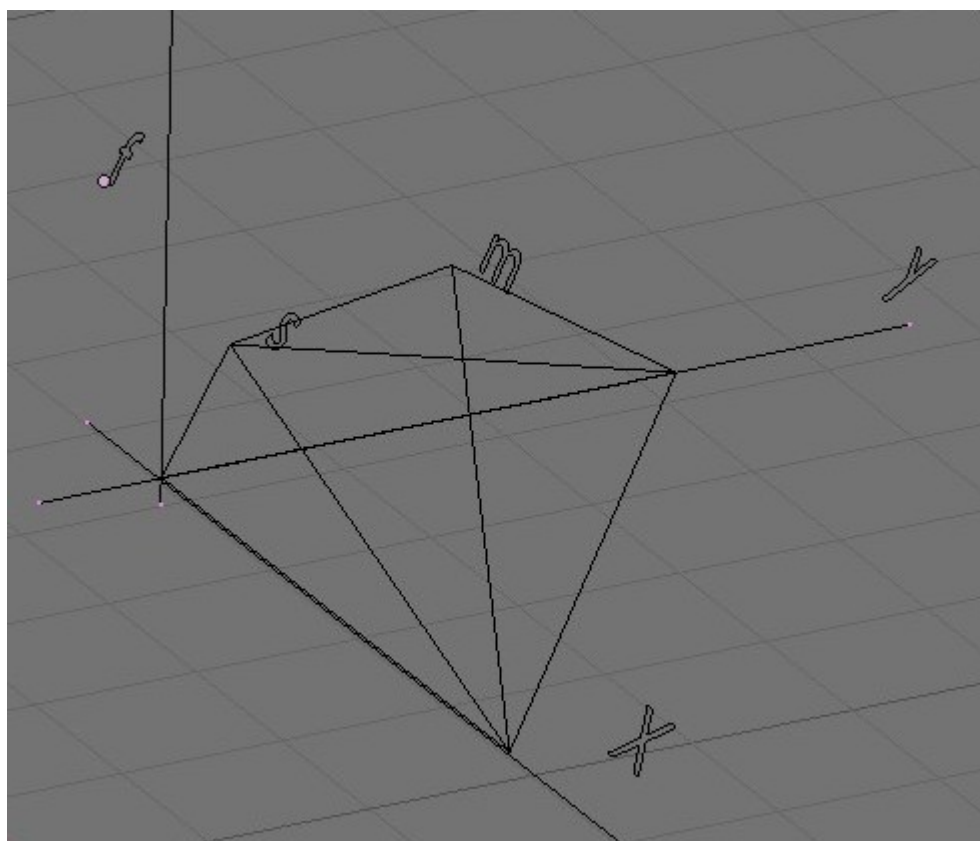


Figure 20: convex function $f(x, y)$.

If we are at the starting point $s(x_s, y_s)$, maximizing on each variable will also have no effect, because both $f(x, y_s)$ and $f(x_s, y)$ have reached their maximums at the point s .

6 Bot Development

6.1 Important Factors in Bot Development

6.1.1 Complex Data Structures

A poker strategy is a giant chunk of data. It is very important to have good data formats for it. Standard .NET datatype can easily fail even with relatively modest amounts of data. For instance, binary serialization fails for a tree of 17000 nodes with 200 elements in each node. XML serialization is more reliable but generates huge files with a high percentage of decoration.

6.1.2 Visualization

A bot works with very complex data structures. It is difficult to imagine them, therefore it is difficult to both write and debug programs processing these structures. It is very important to have an ability to visualize these structures, first of all trees.

Graphviz and ZGRViewer are a great help for this task. pkr framework now also has a good support for tree visualization.

6.1.3 Test Data

If a bot loses in a real game it is very difficult to find why. It can be a wrong algorithm, a bug in the program or in the configuration, a wrong environment where the program is running, etc.

Test data allow to quickly verify if everything is correct without having to waste days for computing and testing strategies for real games.

The most important kind of test data are simplified games. Now they can be solved in a matter of minutes with standard verified algorithms and the results can be used to verify algorithms of the bot.

It is also important to unit-test bots, for obvious reasons. Simplified games can be used for that purpose as well.

6.1.4 Reference Data

It would be a great help to have solutions at least for some real poker problems. Even approximated solution would be very useful. For instance, knowing how much you expect to win in heads-up Holdem in position 0 with AKs allows to quickly verify a strategy.

6.2 Tricks and Troubleshooting

6.2.1 Strategy for Single Pocket for an BR Bot

An BR can be computed for each pocket separately. This can be used to speed-up testing of an BR bot. It is faster to both calculate and test a strategy for one pocket. The poker server can be configured to deal only this pocket to the bot under test.

6.2.2 Generated Deals

pkrloggen can be used to generate a set of deals with predefined properties, for instance a single pocket for the bot under test and uniformly distributed cards for the opponent.

6.2.3 Replay

A set of games can be replayed multiple times to debug a bot. ParrotBot can be used to simulate actions of opponents.

7 Appendix

7.1 Typographic Conventions

This document uses the following typographic conventions.

Convention	Description
Monospace	Source code snippet.

Table 3. Typographic Conventions

7.2 Glossary

In contrast to the opinions of many people from the domain of software, the usage of abbreviations is essential in such a complex development as poker. This glossary provides explanations to these abbreviations and other terms widely used in the domain of poker theory and development of poker agents.

Equilibrium	See Nash Equilibrium
Nash equilibrium	A set of strategies in which no player can do better by unilaterally changing his or her strategy. See also http://en.wikipedia.org/wiki/Nash_equilibrium .
BR	Best Response algorithm (see 3)
Static strategy	A strategy of a player that does not change over time
Dominant strategy	A strategy is dominant if, regardless of what any other players do, the strategy earns a player a larger payoff than any other. Hence, a strategy is dominant if it is always better than any other strategy, for any profile of other players' actions. Depending on whether "better" is defined with weak or strict inequalities, the strategy is termed strictly dominant or weakly dominant . If one strategy is dominant, than all others are dominated.
Hero	When discussing a strategy, this is the player we are rooting for. For instance, in the BR this is the player who is trying to find best response against other players (opponents).
CT	Chance tree
AT	Action tree
MC	Monte-Carlo

7.3 Figures

7.4 Tables

Table 1. Influence of different moves on node values.....13

Table 2. Data for move 1.0.....16

Table 3. Typographic Conventions.....28

7.5 Bibliography

- [1] B. Hoehn, F. Southey, R. C. Holte, V. Bulitko, Effective Short-Term Opponent Exploitation in Simplified Poker, [aaai2005poker.pdf](#)
- [2] Kuhn, H. W. 1950. A simplified two-person poker. *Contributions to the Theory of Games* 1:97-103
- [3] A. Gilpin, T. Sandholm, Lossless abstraction of imperfect information games, extensive.[JACM.pdf](#)