An Optimal Algorithm for Calculating the Profit in the Coins in a Row Game

Tomasz Idziaszek

University of Warsaw
idziaszek@mimuw.edu.pl

Abstract. On the table there is a row of n coins of various denominations. Two players are alternately making moves which consist of picking a coin on one end of the row and collecting it. We show an $O(n)$ time algorithm which calculates the profit of each player assuming they both play optimally.
We also consider a generalization of the game, in which there could be more rows and some rows could be replaced by stacks, i.e., the players can pick coins from only one end of a stack. We show an $O(n \log k)$ algorithm which calculates the profit, where n is the total number of coins and k is the total number of rows and stacks.

Keywords: combinatorial game theory, coins in a row game, graph grabbing

1 Introduction

In Peter Winkler's *Mathematical Puzzles* [7] the very first puzzle is the following game played by two players. On the table there is a row of n coins of various denominations. Two players are alternately making moves which consist of picking a coin on one end of the row and collecting it.

The puzzle is to find a strategy for the first player when n is even which secures her at least the same profit as the profit obtained by the second player. The strategy is as follows: the first player can always secure for herself the coins from odd-numbered positions or from even-numbered positions, thus she will choose the variant which gives her bigger profit.

However we want to solve more ambitious task of finding the optimal strategy, i.e., such strategy that secures the maximum possible profit regardless of opponents moves. It is easy to see that even when n is even the above strategy is not optimal.

There is a simple dynamic programming algorithm which calculates the strategy in $O(n^2)$ time. Let $a[1], \ldots, a[n]$ be the coins' denominations. Let $d[i, j]$ be the maximum advantage of the first person to move in the subsequence of coins numbered $i, i+1, \ldots, j$ $(1 \le i \le j \le n)$. The values of the table can be computed as follows:

$$d[i,j] = \begin{cases} a[i] & \text{for } i = j, \\ \max(a[i] - d[i+1,j],\ a[j] - d[i,j-1]) & \text{for } i < j. \end{cases}$$

Therefore as long as both players play optimally, the first one will get an advantage of $d[1,n]$, thus she will secure $(d[1,n] + \sum_{i=1}^{n} a[i])/2$.

The table allows us to make an optimal move in constant time: if the remaining coins are numbered $i, i+1, \ldots, j$ then taking the left (i-th) coin is an optimal move if $a[i] - d[i+1,j] \geq a[j] - d[i,j+1]$. The more interesting question is whether $\Omega(n^2)$ preprocessing is necessary to find the strategy? And what if we would like to solve a simpler problem of calculating the profits in the optimal play (equivalently the value of $d[1,n]$)?

In the paper we present a faster algorithm which computes the profits of players in the optimal play in the optimal time $O(n)$. It is easy to see that using such algorithm we can calculate an optimal move in $O(n)$ time.

In fact the algorithm we present is able to solve more general problem. Instead of only one row we allow more rows, and the move is to pick a coin from one end of one row. This is equivalent to Peter Winkler's Pizza Problem [2] when initially at least one pizza slice is taken. We also allow to change some rows to stacks, i.e., the players are allowed to pick coins from only one side of a stack.

For the generalized version we present an algorithm which computes the profits of players in the optimal play in time $O(n \log k)$ where n is the total number of coins and k is the total number of rows and stacks.

2 Preliminaries

We can reformulate the problem in the language of graph theory. We are given an undirected graph $G = (V, E)$ and a valuation function $f\colon V \to \mathbb{R}$ which assigns to every node v of the graph its value $f(v)$. Two players are alternately making moves. A move consists of removing a leaf v (i.e., a node which has at most one edge adjacent to it) from the graph and increasing the profit of the player by $f(v)$. Some nodes of the graph are said to be *anchored*. Such nodes could be removed only if they are isolated (i.e., no edge is adjacent to them). In this paper we consider a simple class of graphs, such that every connected component in a graph is a path and every component has at most one anchored node which must be a leaf.

Every move is uniquely described by a node which has been removed during the move. The *play* on a graph G is a sequence of subsequent moves $\alpha = v_1 v_2 \ldots v_k$, where $v_1, \ldots, v_k \in V$. The *value of a play* $\alpha \in V^*$ is defined as the difference between the amount collected by the first player and the amount collected by the second player:

$$\mathrm{val}_G(\alpha) := f(v_1) - f(v_2) + \cdots + (-1)^{k+1} f(v_k).$$

The first player tries to maximize the value of a play, whereas the second player tries to minimize it.

A *strategy* in a game G is a function $\sigma\colon V^* \to V$ such that $\sigma(\alpha)$ specifies a node which should be removed if the already removed nodes form a play α. Note that a strategy for the first (second) player need to consider only plays of even (odd) length.

Given strategies σ and ρ of the first and the second player, respectively, we denote by $\alpha(\sigma, \rho) = v_1 v_2 \dots v_n$ the play which is induced by these strategies:

$$v_i = \begin{cases} \sigma(v_1 v_2 \dots v_{i-1}) \text{ for odd } i, \\ \rho(v_1 v_2 \dots v_{i-1}) \text{ for even } i. \end{cases}$$

The *value of a play induced by strategies* σ and ρ is

$$\mathrm{val}_G(\sigma, \rho) := \mathrm{val}_G(\alpha(\sigma, \rho)).$$

We are interested in the *value of a game*, i.e., in the value of an optimal play. This can be defined recursively. Let $\mathrm{val}(G; V_A)$ be the value of a game in the graph G after removing nodes from $V \setminus V_A$ and let $\mathrm{Avail}(G; V_A)$ be the set of available nodes. Then

$$\mathrm{val}(G; V_A) = \begin{cases} 0 & \text{if } V_A = \emptyset, \\ \max_{v \in \mathrm{Avail}(G;V_A)} f(v) - \mathrm{val}(G; V_A \cup \{v\}) & \text{otherwise.} \end{cases}$$

The value of a game is

$$\mathrm{val}(G) := \mathrm{val}(G; V)$$

and the profit of the first player is

$$\Big(\mathrm{val}(G) + \sum_{v \in V} f(v)\Big)/2.$$

The value of a game can be also stated in terms of strategies:

$$\mathrm{val}(G) = \max_{\sigma} \min_{\rho} \mathrm{val}_G(\sigma, \rho).$$

It follows that for every first-player strategy σ,

$$\mathrm{val}(G) \geq \min_{\rho} \mathrm{val}_G(\sigma, \rho) \tag{1}$$

and for every second-player strategy ρ, $\mathrm{val}(G) \leq \max_{\sigma} \mathrm{val}_G(\sigma, \rho)$.

A first-player strategy σ is *optimal* if

$$\mathrm{val}(G) = \min_{\rho} \mathrm{val}_G(\sigma, \rho) \tag{2}$$

which means that for every second-player strategy ρ,

$$\mathrm{val}(G) \leq \mathrm{val}_G(\sigma, \rho). \tag{3}$$

Note that an optimal strategy does not need to obtain the best possible profit for the first player, it only need to obtain the advantage of at least $\mathrm{val}(G)$. Similarly a second-player strategy ρ is optimal if $\mathrm{val}(G) = \max_{\sigma} \mathrm{val}_G(\sigma, \rho)$ which means that for every first-player strategy σ, $\mathrm{val}(G) \geq \mathrm{val}_G(\sigma, \rho)$.

Three proofs in this paper will follow the following setting. We consider an optimal first-player strategy σ and we want to develop another strategy σ' which will be also optimal but with additional properties, or better than σ, thus it will lead to a contradiction. In order to show correspondence between these two strategies, we need to show how σ' performs against *each possible* second-player strategy ρ. The strategy σ' is constructed based on the play of σ against a

certain second-player strategy ρ'. The following lemma shows the correspondence between the strategies:

Lemma 1. *Let σ be an optimal first-player strategy and σ' be an first-player strategy. Then:*

(i) exists second-player strategy ρ such that for every second-player strategy ρ', $\mathrm{val}_G(\sigma, \rho') \geq \mathrm{val}_G(\sigma', \rho)$,
(ii) σ' is optimal if and only if for every second-player strategy ρ exists such second-player strategy ρ' that $\mathrm{val}_G(\sigma, \rho') \leq \mathrm{val}_G(\sigma', \rho)$.

Proof. (i) Since σ is optimal then from (3) and (1),

$$\mathrm{val}_G(\sigma, \rho') \geq \mathrm{val}(G) \geq \min_{\rho} \mathrm{val}_G(\sigma', \rho).$$

It suffices to take as ρ the strategy which yields the minimum.

(ii) The "only if" part follows directly from (i) by changing the roles of σ and σ'. For the "if" part observe that from the statement we have

$$\min_{\rho'} \mathrm{val}_G(\sigma, \rho') \leq \min_{\rho} \mathrm{val}_G(\sigma', \rho)$$

Since σ is optimal then from (2) and (1),

$$\mathrm{val}(G) = \min_{\rho'} \mathrm{val}_G(\sigma, \rho') \leq \min_{\rho} \mathrm{val}_G(\sigma', \rho) \leq \mathrm{val}(G).$$

Thus from (2), σ' is optimal. □

Based on strategies σ and ρ we construct a desired strategy σ' and an auxiliary strategy ρ'. Sometimes it will be sufficient to simply copy moves from the given strategies. Let $\alpha = v_1 v_2 \dots v_k$ be a play of σ against ρ' and $\alpha' = v'_1 v'_2 \dots v'_k$ be a play of σ' against ρ. We say that we *echo* in the i-th move ($1 \leq i \leq k$) if

$$v_i = v'_i$$

which means that for odd i we set $\sigma'(v'_1 \dots v'_{i-1})$ based on $\sigma(v_1 \dots v_{i-1})$ and for even i we set $\rho'(v_1 \dots v_{i-1})$ based on $\rho(v'_1 \dots v'_{i-1})$. It is easy to see that if we echo in all moves from i-th to k-th, then

$$\mathrm{val}_G(v'_1 \dots v'_k) - \mathrm{val}_G(v_1 \dots v_k) = \mathrm{val}_G(v'_1 \dots v'_{i-1}) - \mathrm{val}_G(v_1 \dots v_{i-1}).$$

3 Overview of the Algorithm

To get some intuition behind the algorithm presented in the paper, let us consider the node M in the graph which has the greatest value $f(M)$ among all nodes. Since both players want to maximize their profit, they are interested in removing nodes of high value, so the node M is especially attractive for them. It turns out that in case of the greatest value it pays off to be greedy:

Theorem 1 (Greedy Move Principle). *Let M be an available leaf and $f(M)$ be no smaller than any value in the graph G. Let G' be a graph which is formed from G by deleting the leaf M. Then* $\mathrm{val}(G) = f(M) - \mathrm{val}(G')$.

Unfortunately, for most of the time the players will not be lucky enough to apply Theorem 1 directly. If the node M is not an available leaf, then at some point in the game one of the players makes a move which uncovers it. Then the opponent will take this node, which is a loss for the first player. Thus if the node M was the last one in its component then such a move was fruitless for the first player, so she should avoid such moves as long as possible. In the other case, the only reason to make such move was a desire to remove a node which was uncovered after removing M. This reasoning leads to the following two theorems:

Theorem 2 (Fusion Principle). *Let x, M, y be adjacent nodes in the graph G such that $f(x), f(y) \leq f(M)$. Let G' be a graph formed from G by fusing these three nodes into a single node v of value $f(v) = f(x) - f(M) + f(y)$, which is anchored if x or y was anchored. Then* $\mathrm{val}(G) = \mathrm{val}(G')$.

Theorem 3 (Fruitless Move Principle). *Let x, M be adjacent nodes in the graph G of n nodes such that M is anchored and $f(x) \leq f(M)$. Let G' be a graph formed from G by deleting these two nodes, and making the potential another neighbour of x anchored. Then* $\mathrm{val}(G) = \mathrm{val}(G') + (-1)^n(f(x) - f(M))$.

The next section of the paper is devoted to proving these three theorems. However, since they are sufficient for constructing an optimal algorithm for calculating the profits, we begin with presentation of the algorithm.

In the first step of the algorithm we replace each component of the graph with an equivalent one (i.e., such that does not alter the value of the game) on which the value function is bitonic (i.e., considering the nodes in the order they appear in the component, the value function is decreasing up to some node, and then it is increasing). We do this by applying the Fusion Principle as long as it is possible. We can do it in $O(n)$ time by pushing the subsequent values on a stack and checking whether we can do fusion on the top three values from the stack. In the next step we apply the Fruitless Move Principle as long as it is possible for every component which has an anchored node. This step is easily done in $O(n)$ time.

At this point we can apply the Greedy Move Principle till the end of the game, since regardless of the moves, a node with the greatest value will always be available. Therefore in the optimal play the nodes will be removed in the order of their values, thus we just sort all values of the nodes, which can be done in $O(n \log k)$ time, where k is the number of available leafs, by merging k sorted lists.

The pseudocode for the algorithm is presented in the Appendix.

Theorem 4. *There is an $O(n)$ time algorithm for calculating the value of the game in the Coins in a Row game. There is an $O(n \log k)$ time algorithm for calculating the value of the game in the generalized Coins in a Row game, where k is the number of available leafs.*

The names of the Theorems 1, 2, and 3 were inspired by the names used in [1] for Green Hackenbush.

4 Proofs of the Principles

In this section we prove Theorems 1, 2 and 3. First we prove the Greedy Move Principle. Then we prove the Fusion Principle and the Fruitless Move Principle in the special case when the node M has the greatest value in the graph (but there can be other nodes in the graph with the same greatest value). Finally, we prove these principles in the general case.

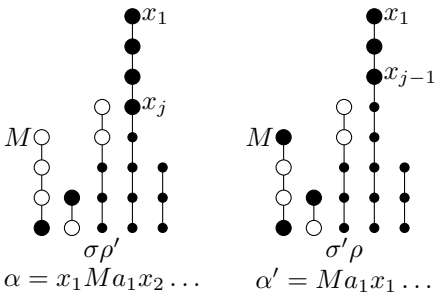
4.1 The Greedy Move Principle

Lemma 2. *Let M be an available leaf and $f(M)$ be the unique greatest value in the graph. Then every optimal first-player strategy removes M in the first move.*

Proof. Let σ be a optimal first-player strategy and assume by contradiction that $\sigma(\epsilon) = x_1 \neq M$. We construct a strategy σ' such that $\sigma'(\epsilon) = M$, which will do better than σ. For this we consider any second-player strategy ρ which will play against σ'. In order to construct σ' we play a strategy σ against ρ' such that $\rho'(x_1) = M$.

Let $x_1, x_2, \ldots$ be nodes in the connected component of the leaf x_1 in the order of their removal.

On the following figure the nodes removed by the first player are marked with black circles and the nodes removed by the second player are marked with white circles.



Let $\alpha = x_1 M v_1 v_2 \ldots v_{2k}$ be a play in σ against ρ', whereas $\alpha' = M v'_1 v'_2 \ldots v'_{2k}$ be a play in σ' against ρ. The following invariant holds during the first phase of the plays: for every $0 \leq i < k$,

$$v'_{2i+1} = v_{2i+2},$$
$$v'_{2i+2} = v_{2i+1} \text{ or } (v'_{2i+2} = x_j \text{ and } v_{2i+1} = x_{j+1} \text{ for some } j).$$

It follows that there exists j such that $x_1, \ldots, x_j \in \alpha$, $x_{j+1} \notin \alpha$ and

$$\mathrm{val}_G(\alpha') - \mathrm{val}_G(\alpha) = 2f(M) - f(x_j).$$

At this time the opponent makes a move $v'_{2k+1} := \rho(\alpha')$. We consider two cases.

Case (1) There is $v'_{2k+1} = x_j$. It causes that the sets of vertices in both plays α, $\alpha' v'_{2k+1}$ to be equal and $\mathrm{val}_G(\alpha') - \mathrm{val}_G(\alpha) = 2f(M) - 2(x_j)$. From now on we will echo the moves which will ensure that $\mathrm{val}_G(\rho', \sigma) - \mathrm{val}_G(\rho, \sigma) = 2f(M) - 2f(x_j)$. Since $f(M) > f(x_j)$ then $\mathrm{val}_G(\rho', \sigma) > \mathrm{val}_G(\rho, \sigma')$.

Case (2) In the other case we look at $v_{2k+1} := \sigma(\alpha)$.

(i) If $v_{2k+1} = v'_{2k+1}$, then we put $\sigma'(\alpha' v'_{2k+1}) := x_j$. Again the sets α and $\alpha' v'_{2k+1}$ are equal and $\mathrm{val}_G(\alpha') - \mathrm{val}_G(\alpha) = 2f(M) - 2(v_{2k+1}) > 0$. Again we echo the moves and get $\mathrm{val}_G(\rho', \sigma) > \mathrm{val}_G(\rho, \sigma')$.

(ii) If $v_{2k+1} = x_{j+1}$ then we put $\sigma'(\alpha' v'_{2k+1}) := x_j$ and $\rho'(\alpha v_{2k+1}) := v'_{2k+1}$.

(iii) Otherwise we put $\sigma'(\alpha' v'_{2k+1}) := v_{2k+1}$ and $\rho'(\alpha v_{2k+1}) := v'_{2k+1}$.

In some point in the play we will have (1) or (2i) and thus $\mathrm{val}_G(\rho', \sigma) > \mathrm{val}_G(\rho, \sigma')$. Since it holds for every second-player strategy ρ, it contradicts the first point of Lemma 1. Thus σ is not optimal. □

Proof (of the Greedy Move Principle). From Lemma 2 removing M if $f(M)$ is unique is the only winning move. Using the same reasoning as in Lemma 2 one can prove that removing M if $f(M)$ is not unique is a winning move. □

4.2 The Fusion Principle for $f(M)$ Being the Greatest Value

For a given node M we say that a strategy is *M-greedy* if it removes the node M as soon as it is available.

Lemma 3. *Let x, M, y be adjacent nodes and $f(M)$ be the greatest value in the graph. There is an optimal first-player strategy σ' which during a play against every M-greedy second-player strategy ρ when removes the node x in the move i, it removes the node y in the move $i + 2$.*

Proof. Let σ be any optimal first-player strategy. As long as σ does not remove the node x or the node y we echo the strategies. Thus without lose of generality we can assume that $\sigma(\epsilon) = x$. We put $\rho'(x) = M$.

Again we echo the strategies until either σ removes y or ρ removes x or y. Thus we have a play $\alpha = xMv_1v_2 \dots v_k$ of σ against ρ' and a play $\alpha' = v_1v_2 \dots v_k$ of σ' against ρ. Moreover $\mathrm{val}_G(\alpha') - \mathrm{val}_G(\alpha) = f(M) - f(x)$.

In the first case we have k even and $\sigma(\alpha) = y$. Then we put $\sigma'(\alpha') := x$. Now from M-greediness of ρ we have $\rho(\alpha' x) = M$. Finally we put $\sigma'(\alpha' xM) := y$ to satisfy the property of σ' from the statement. At this moment sets of nodes in αy and $\alpha' xMy$ are equal and $\mathrm{val}_G(\alpha' xMy) - \mathrm{val}_G(\alpha y) = 0$.

In the second case we have k odd and $\rho(\alpha') = x$. We put $\sigma'(\alpha' x) := M$. Now sets of nodes in α and $\alpha' xM$ are equal and $\mathrm{val}_G(\alpha' xM) - \mathrm{val}_G(\alpha) = 2f(M) - 2f(x) > 0$.

In the third case we have k odd and $\rho(\alpha') = y$. We put $\rho'(\alpha) := y$ and $\sigma'(\alpha' y) := x$. Now from M-greediness of ρ we have $\rho(\alpha' yx) = M$. At this moment sets of nodes in αy and $\alpha' yxM$ are equal and $\mathrm{val}_G(\alpha' yxM) - \mathrm{val}_G(\alpha y) = 0$.

Next we echo moves till the end of a game. Thus $\mathrm{val}_G(\sigma', \rho) - \mathrm{val}_G(\sigma, \rho') \geq 0$ and from Lemma 1 we have that σ' is optimal. □

Lemma 4. *Let x, M, y be adjacent nodes and $f(M)$ be the greatest value in the graph G. Let G' be a graph formed from G by fusing these three nodes into a single node v with $f(v) = f(x) - f(M) + f(y)$, which is anchored if x or y was anchored. Then* $\mathrm{val}(G) = \mathrm{val}(G')$.

Proof. Let σ be an optimal first-player strategy in G which is M-greedy and satisfies the property of Lemma 3. Now we show the strategy σ' in G'. Again σ' plays against every strategy ρ and uses σ which plays against ρ' in G.

The strategy σ' echoes with one special case. When σ removes x (or y) then σ' removes v and orders ρ' to remove M (thus ρ' is M-greedy), and then from the assumption σ will remove y (or x).

The strategy ρ' also echoes with one special case. When ρ removes v then ρ' removes x (or y, whichever will be available). Then from the M-greediness σ removes M and then ρ' responds by removing y (or x respectively).

Therefore we get

$$\mathrm{val}_G(\sigma, \rho') = \mathrm{val}_{G'}(\sigma', \rho)$$

Let ρ^* be a strategy which yields minimum in (1). Then from (1) and from optimality of σ and (2),

$$\mathrm{val}(G) \leq \mathrm{val}_G(\sigma, \rho') = \mathrm{val}_{G'}(\sigma', \rho^*) \leq \mathrm{val}(G'),$$

thus $\mathrm{val}(G) \leq \mathrm{val}(G')$.

Now consider a graph G^* formed from G by adding a single isolated node M^* which value $f(M^*)$ is greater than any other value in G. Similarly consider G'^*. From Theorem 1 we get

$$\mathrm{val}(G^*) = f(M^*) - \mathrm{val}(G), \qquad \mathrm{val}(G'^*) = f(M^*) - \mathrm{val}(G').$$

Using the same reasoning as above we also get that $\mathrm{val}(G^*) \leq \mathrm{val}(G'^*)$, since after removing M^* in the first move, the value $f(M)$ will become the greatest one in G^* and we can apply the assumption of M-greediness. Therefore

$$f(M^*) - \mathrm{val}(G) = \mathrm{val}(G^*) \leq \mathrm{val}(G'^*) = f(M^*) - \mathrm{val}(G'),$$

thus $\mathrm{val}(G) \geq \mathrm{val}(G')$. □

4.3 The Fruitless Move Principle for $f(M)$ Being the Greatest Value

Lemma 5. *Let x, M be adjacent nodes such that M is anchored and $f(M)$ be the greatest value in G. There is an optimal first-player strategy σ' which does not remove x, unless only x and M are left in the graph.*

Proof. Let σ be any optimal first-player strategy. As long as σ does not remove the node x we echo the strategies. Thus without lose of generality we can assume that $\sigma(\epsilon) = x$. We put $\rho'(x) = M$.

Again we echo the strategies until either G is empty when it is σ's turn or ρ removes x. Thus we have a play $\alpha = xMv_1v_2 \dots v_k$ of σ against ρ' and a play $\alpha' = v_1v_2 \dots v_k$ of σ' against ρ. Moreover $\mathrm{val}_G(\alpha') - \mathrm{val}_G(\alpha) = f(M) - f(x)$.

In the first case we have k even and G is empty. Then we put $\sigma'(\alpha') := x$. Now there is only one move of $\rho(\alpha' x) = M$. At this moment sets of nodes in α and $\alpha' x M$ are equal and $\mathrm{val}_G(\alpha' x M) - \mathrm{val}_G(\alpha) = 0$.

In the second case we do exactly the same as we did in the second case in Lemma 3.

Next we echo moves till the end of a game. Thus $\mathrm{val}_G(\sigma', \rho) - \mathrm{val}_G(\sigma, \rho') \geq 0$ and from Lemma 1 we have that σ' is optimal. □

Lemma 6. *Let x, M be adjacent nodes such that M is anchored and $f(M)$ be the greatest value in the graph G with n nodes. Let G' be a graph which is formed from G by deleting these two nodes, and making the potential another neighbour of x anchored. Then* $\mathrm{val}(G) = \mathrm{val}(G') + (-1)^n(f(x) - f(M))$.

Proof. Let σ be an optimal first-player strategy in G which satisfies the property of Lemma 5. We can use it in G' ignoring the potential last move of σ which removes x. Observe that whenever σ plays in G then the first-player will not remove the node x, unless n is even. Thus for odd n,

$$\mathrm{val}(G) \leq \mathrm{val}(G') - f(x) + f(M),$$

whereas for even n,

$$\mathrm{val}(G) \leq \mathrm{val}(G') + f(x) - f(M) \leq \mathrm{val}(G') - f(x) + f(M),$$

thus $\mathrm{val}(G) \leq \mathrm{val}(G') + (-1)^n(f(x) - f(M))$.

Let σ' be an optimal first-player strategy in G'. We can use it in G with additional requirement that if the opponent removed x we remove M. Again σ' can be forced to remove the node x in G only if n is even, thus following the above reasoning, $\mathrm{val}(G) \geq \mathrm{val}(G') + (-1)^n(f(x) - f(M))$. □

4.4 The Fusion Principle and the Fruitless Move Principle

Finally, we are ready to prove Theorems 2 and 3.

Proof (of the Fusion Principle and the Fruitless Move Principle). We prove the both theorems simultaneously by induction on the number of nodes which have strictly greater values than $f(M)$. Induction base (when there is no node with greater value than $f(M)$) follows immediately from Lemmas 4 and 6.

By *reduction* we name an operation of fusing nodes x, M, y into one node if the assumptions of the Fusion Principle are satisfied or of deleting nodes x, M if the assumptions of the Fruitless Move Principle are satisfied. We say that G *reduces* to G'.

Assume that the greatest value is assigned to the node N, $f(N) > f(M)$. Observe that since $f(x), f(y) \leq f(M)$ then the node N is different from x and y. We consider five cases. In every case we will construct a graph G_N from G and a graph G'_N from G' such that there is a smaller number of nodes of values greater than $f(M)$ in G_N (G'_N) than in G (G'). Moreover, G_N will reduce to G'_N, thus we will be able to apply the inductive assumption on G_N and G'_N.

Case (1) N is an available leaf. Let G_N (G'_N) be formed from G (G') by removing the node N. From Theorem 1,

$$\mathrm{val}(G) = f(N) - \mathrm{val}(G_N), \quad \mathrm{val}(G') = f(N) - \mathrm{val}(G'_N).$$

From the inductive assumption,

$$\mathrm{val}(G_N) = \mathrm{val}(G'_N) \quad \text{or} \quad \mathrm{val}(G_N) = \mathrm{val}(G'_N) + (-1)^{n-1}(f(x) - f(M)).$$

depending on the type of the reduction. In case of the fusion reduction

$$\mathrm{val}(G) = f(N) - \mathrm{val}(G_N) = f(N) - \mathrm{val}(G'_N) = \mathrm{val}(G')$$

and in case of the fruitless reduction

$$\begin{aligned}\mathrm{val}(G) &= f(N) - \mathrm{val}(G_N) = f(N) - \mathrm{val}(G'_N) - (-1)^{n-1}(f(x) - f(M)) = \\ &= \mathrm{val}(G') + (-1)^n(f(x) - f(M)).\end{aligned}$$

Case (2) N is not a leaf and it is not adjacent to neither x nor y. Let x' and y' be nodes adjacent to N. Let G_N (G'_N) be formed from G (G') by fusing the nodes x', N, y' into a node w of value $f(w) = f(x') - f(N) + f(y')$. From Lemma 4,

$$\mathrm{val}(G) = \mathrm{val}(G_N), \quad \mathrm{val}(G') = \mathrm{val}(G'_N).$$

From the inductive assumption,

$$\mathrm{val}(G_N) = \mathrm{val}(G'_N) \quad \text{or} \quad \mathrm{val}(G_N) = \mathrm{val}(G'_N) + (-1)^{n-2}(f(x) - f(M)).$$

Now we use the same reasoning as in the case (1).

Case (3) N is an anchored leaf and it is not adjacent to neither x nor y. Let x' be a node adjacent to N. Let G_N (G'_N) is formed from G (G') by deleting x' and N. From Lemma 6,

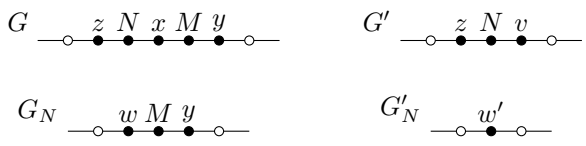
$$\mathrm{val}(G) = \mathrm{val}(G_N) + (-1)^n(f(x') - f(N)), \quad \mathrm{val}(G') = \mathrm{val}(G'_N) + (-1)^n(f(x') - f(N)).$$

From the inductive assumption,

$$\mathrm{val}(G_N) = \mathrm{val}(G'_N) \quad \text{or} \quad \mathrm{val}(G_N) = \mathrm{val}(G'_N) + (-1)^{n-2}(f(x) - f(M)).$$

We use the same reasoning as in the case (1).

Case (4) N is not a leaf and without losing of generality it is adjacent to x. Firstly, we consider the case of the fusion reduction (see figure).



In the graph G we have five adjacent nodes z, N, x, M, y and in the graph G' we have adjacent nodes z, N, v where $f(v) = f(x) - f(M) + f(y) \leq f(M) < f(N)$. Let G_N be formed from G by fusing the nodes z, N, x in one node w of value

$f(w) = f(z) - f(N) + f(x)$. Let G'_N be formed from G' by fusing the nodes z, N, v in one node w' of value $f(w') = f(z) - f(N) + f(v)$. We can apply Lemma 4 to get

$$\mathrm{val}(G) = \mathrm{val}(G_N), \quad \mathrm{val}(G') = \mathrm{val}(G'_N)$$

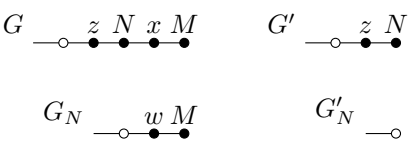
In G_N we have w, M, y and in G'_N we have w'. Since

$$f(w') = f(z) - f(N) + f(x) - f(M) + f(y) = f(w) - f(M) + f(y)$$

and $f(w) \leq f(x) \leq f(M)$ then we can apply the inductive assumption and get

$$\mathrm{val}(G_N) = \mathrm{val}(G'_N).$$

Now we consider the case of the fruitless reduction (see figure).



In the graph G we have four adjacent nodes z, N, x, M where M is anchored and in the graph G' we have adjacent nodes z, N where N is anchored. Let G_N be formed from G by fusing the nodes z, N, x in one node w of value $f(w) = f(z) - f(N) + f(x) \leq f(x) \leq f(M)$. Let G'_N be formed from G' by deleting the nodes z and N. We can apply Lemmas 4 and 6 to get

$$\mathrm{val}(G) = \mathrm{val}(G_N), \quad \mathrm{val}(G') = \mathrm{val}(G'_N) + (-1)^{n-2}(f(z) - f(N))$$

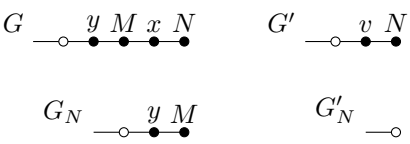
From the inductive assumption,

$$\mathrm{val}(G_N) = \mathrm{val}(G'_N) + (-1)^{n-2}(f(w) - f(M)),$$

thus

$$\begin{aligned}\mathrm{val}(G) = \mathrm{val}(G_N) &= \mathrm{val}(G'_N) + (-1)^{n-2}(f(z) - f(N) + f(x) - f(M)) = \\ &= \mathrm{val}(G') + (-1)^n(f(x) - f(M)).\end{aligned}$$

Case (5) N is an anchored leaf and without losing of generality it is adjacent to x. We cannot have fruitless reduction, since then both N and M would be anchored which is not possible. Thus we only consider fusion reduction (see figure).



In the graph G we have four adjacent nodes N, x, M, y where N is anchored and in the graph G' we have adjacent nodes N, v where N is anchored. Let G_N be formed from G by deleting the nodes N and x. Let G'_N be formed from G' by deleting the nodes N and v. From the Lemma 6 we get

$$\mathrm{val}(G) = \mathrm{val}(G_N) + (-1)^n(f(x) - f(N)), \quad \mathrm{val}(G') = \mathrm{val}(G'_N) + (-1)^{n-2}(f(v) - f(N)).$$

From the inductive assumption,

$$\mathrm{val}(G_N) = \mathrm{val}(G'_N) + (-1)^{n-2}(f(y) - f(M)),$$

thus

$$\begin{aligned}\mathrm{val}(G) &= \mathrm{val}(G_N) + (-1)^n(f(x) - f(N)) = \\ &= \mathrm{val}(G'_N) + (-1)^n(f(x) - f(M) + f(y) - f(N)) = \mathrm{val}(G').\end{aligned}$$

□

5 Conclusions

There are several open problems related to the Coins in a Row game. We presented an optimal algorithm for calculating the value of the game, but there is still open question of an algorithm which calculates the optimal moves for one player during the whole play in total time $o(n^2)$, which would be better than naïve dynamic programming algorithm. Also there is a question whether the algorithm could be used for computing the first move in the Peter Winkler's Pizza Problem when initially the whole pizza is intact in time $o(n^2)$.

The game could be generalized in various directions. One possible direction is to allow more general class of graphs, such as trees [5], unrooted or rooted (i.e., with one anchored node), or forests. The tools developed in this paper look promising in developing the polynomial time algorithm here, e.g., the proof of the Greedy Move Principle holds, the Fusion Principle should still be true, and preliminary research show that the Fruitless Move Principle should be generalizable for rooted trees. The solution for unrooted trees can be reduced to rooted trees as shown by Seacrest and Seacrest [6].

In order to allow any graph we should rephrase the definition of a move. We say that removing a node is possible if the number of connected components in a graph does not increase [4]. Cibulka et al. [3] showed that the problem is PSPACE-complete even for connected graphs with no anchored nodes.

References

1. E.R. Berlekamp, J.H. Conway, and R.K. Guy. *Winning Ways for Your Mathematical Plays.* A.K. Peters, 2004.
2. Josef Cibulka, Jan Kyncl, Viola Mészáros, Rudolf Stolar, and Pavel Valtr. Solution of peter winkler's pizza problem. *CoRR*, abs/0812.4322, 2008.
3. Josef Cibulka, Jan Kyncl, Viola Mészáros, Rudolf Stolar, and Pavel Valtr. Graph sharing games: Complexity and connectivity. In Jan Kratochvíl, Angsheng Li, Jirí Fiala, and Petr Kolman, editors, *TAMC*, volume 6108 of *Lecture Notes in Computer Science*, pages 340–349. Springer, 2010.
4. Piotr Micek and Bartosz Walczak. A graph-grabbing game. *Combinatorics, Probability & Computing*, 20(4):623–629, 2011.
5. Moshe Rosenfeld. A gold-grabbing game, `http://garden.irmacs.sfu.ca/?q=op/a_gold_grabbing_game`.
6. Deborah E. Seacrest and Tyler Seacrest. Grabbing the gold. 2010.
7. Peter Winkler. *Mathematical Puzzles: A Connoisseur's Collection.* AK Peters, 2004.

A Pseudocode of the algorithm

$val \leftarrow 0$
for every connected component c in the graph G **do**
  $m_c \leftarrow 0$
  for every subsequent node v in c **do**
    $m_c \leftarrow m_c + 1$
    $s_c[m_c] \leftarrow f(v)$
    while $m_c \geq 3$ **and** $s_c[m_c - 2] \leq s_c[m_c - 1]$ **and** $s_c[m_c - 1] \geq s_c[m_c]$ **do**
      $s_c[m_c - 2] \leftarrow s_c[m_c - 2] - s_c[m_c - 1] + s_c[m_c]$ {*Fusion Principle*}
      $m_c \leftarrow m_c - 2$
  {*we assume that if a component has an anchored node that it is the last one*}
  if the last node in c is anchored **then**
    while $m_c \geq 2$ **and** $s_c[m_c - 1] \leq s_c[m_c]$ **do**
      $val \leftarrow val + (-1)^n (s_c[m_c - 1] - s_c[m_c])$ {*Fruitless Move Principle*}
      $m_c \leftarrow m_c - 2$
$S \leftarrow$ multiset of all values from $s_c[1..m_c]$ for all components c
$sign \leftarrow 1$
for every x in S in nonincreasing order **do**
  $val \leftarrow val + sign \cdot x$ {*Greedy Move Principle*}
  $sign \leftarrow -sign$
return val