Winning Strategies and Complexity of Nim-Type Computer Game on Plane

Boris S. Verkhovsky
Computer Science Department, College of Computing Sciences
New Jersey Institute of Technology, Newark, USA
E-mail: verb@njit.edu
Received June 10, 2010; revised July 17, 2010; accepted August 22, 2010

Abstract

A Nim-type computer game of strategy on plane is described in this paper. It is demonstrated that winning strategies of this two-person game are determined by a system of equations with two unknown integer sequences. Properties of winning points/states are discussed and an O(loglogn) algorithm for the winning states is provided. Two varieties of the Game are also introduced and their winning strategies are analyzed.

Keywords: Nim-type Game, Two-person Strategy Game, Winning Strategies, Newton Algorithm, Fibonacci Numbers

1. Introduction

A Nim game is probably one of the most ancient of all known games. There are several varieties of Nim: categorical games in which no draw is possible; futile games which permit a tie (draw); Grundy’s game is a special type of Nim. The game is played by the following rules: given a heap of size \( n \), two players alternately select a sub-heap and divide it into two unequal parts. A player loses if he or she cannot make a legal move. The Misere form of Nim is a version in which the player taking the last piece is the winner, [1].

In Fibonacci Nim, two players deal with a pile of \( n \) stones, where \( n>1 \). The first player may remove any number of stones, provided that at least one stone is left. Players alternate moves under the condition that if one player removed \( x \) stones, then another one may remove at most \( 2x \) stones. Some of them are described in [1-5].

Several years ago the author of this paper introduced a Nim game with a heap of \( N \) stones, where each player is allowed to take at most \( m \) stones, provided that he/she does not repeat the last move of her/his opponent (“do not be a copycat”). The player taking the last stone is the winner. However, a player loses if he/she cannot make a feasible move. Winning strategies for an arbitrary \( m>1 \) were provided by the author of this paper and implemented in [6] and [7] by his graduate students.

In the late 1980’s the author also introduced a variety of the Nim-game that is discussed in this paper. In the paper we study properties of winning points, provide an algorithm for direct computation of winning points and analyze its complexity. It is demonstrated that the algorithm has O(loglogn) time complexity and does not require any storage, save a couple of numbers that are pre-computed at the beginning of the game. Preliminary results of this paper are published in [15].

2. Two-player Game on Plane

1) The Game starts after two distinct non-negative integers \((S_0, L_0)\) are selected randomly; here,

\[
1 \leq p \leq S_0 < qS_0 \leq L_0 ; S := S_0 ; L := L_0 ;
\]  

Remark 1: In the following discussion \((S, L)\) is a point on a two-dimensional plane with integer coordinates; all further points are located in the positive quadrant of the plane; \( p \) and \( q \) determine a “level” of the Game. It is assumed that \( 0 \leq S < L \) holds, otherwise we swap the coordinates.

2) Three types of moves that allowed are: horizontal, vertical and diagonal.

The players on their move may decrease either

a). The first coordinate on an integer \( t \), \((S, L) \rightarrow (S - t, L), \)
\((\text{horizontal move, } h\text{-move, for short})\) or

b). The second coordinate on an integer \( u \),
\((S, L) \rightarrow (S-L-u), \) \{vertical move, v-move, for short\} or c). Both coordinates on the same integer \(x\), 
\((S, L) \rightarrow (S-x, L-x), \) \{diagonal move, d-move, for short\};

The first player that reaches \((0,0)\)-point on her/his move is the winner of the Game. An analogous Nim game was introduced by Wythoff [14]. Wythoff’s game is played with two heaps of counters: a player is allowed to take any number from either heap or the same number from both. The player taking the last counter wins.

As in every two-person game with complete information, this Game has a winning strategy for one of the players [8-10]. In the following discussion we consider that a Human (Hugo) plays against a Computer (Cora).

All points can be divided onto two classes: winning points for Cora and losing points for Cora. It is clear that a winning point for Cora is a losing point for Hugo, and vice versa.

**Definition 1**: We will say that the Game is in a winning state if after Cora’s move it is in a winning point.

Let’s denote Cora’s winning states as \(w_n\), for \(n \geq 0\). Here \(w_0 = (0, 0)\).

**Example 1**: The \(w_1 = (1, 2)\) point is a Cora’s winning point, because Hugo cannot reach \(w_0 = (0, 0)\) point on his move. The \(w_2 = (3, 5)\) is another winning point for Cora, because on his move Hugo cannot reach either \(w_0 = (0, 0)\) point or \(w_1 = (1, 2)\). On the other hand, after any move by Hugo, Cora reaches either \((0,0)\) or \((1,2)\).

### 3. Seven Properties of Winning Points

**P1.** It is easy to see that if \((c, d)\) is a winning point, then \((d, c)\) is also a winning point.

**P2.** With exception of the \((0,0)\)-point, in all other winning points \(c \neq d\). Indeed, let \((c, c)\) be a winning point for Cora. Then Hugo can reach the \((0,0)\)-point using a diagonal move, \{via subtracting the same \(y = c\) from both coordinates\}.

**P3.** If the Game is in a winning point \(w\) after Cora’s move, then there is no move by which Hugo can reach another winning state \(w’\). On the other hand, if the Game is in a losing point \(l\), then there exists at least one move that transforms the Game into a winning state. For example, if after Hugo’s move the Game is in the state \((7, 9)\), then there are two winning moves for Cora: \((4,7)\) and \((3,5)\).

In general, let \(W\) be a set of all winning points and \(L\) be a set of all losing points. Then after one move the Game is transformed from \(W\) to \(L\). However, if the Game is in \(L\), then there exists at least one move that transforms the game into \(W\). Formally it means that, if \((S, L) \in \mathcal{W}\), then for any positive integer \(u\), \((S-u, L) \in \mathcal{W}\) or \((S, L-u) \not\in \mathcal{W}\) or \((S-u, L-u) \not\in \mathcal{W}\).

**P4. Proposition 1**: There are no two winning points \(w_i = (c_i, d_i)\) and \(w_k = (c_k, d_k)\) such that
\[
d_i - c_i = d_k - c_k = r
\]
where \(r\) is an integer.

**Proof**: Let’s assume that for \(i < k\) Equation (2) holds and \(v := d_i - d_i\). Then after Cora’s move \(w_i\) Hugo can reach \(w_i\) via a diagonal move, i.e., by subtracting from both coordinates the same integer \(v\).

**P5. Let** \(w_i\) and \(w_k\) **be two distinct winning points. Then all \(c_i, d_i, c_k, d_k\) are distinct integers, otherwise Hugo would be able to transform the Game into another winning state.**

**P6. Proposition 2**: Let \((S, L) \in \mathcal{W}\). Then for every \(n=1,2,\ldots\) holds
\[
d_n - c_n = n
\]

**Proof**: Let’s assume that there exists at least one winning point \((c_m, d_m)\), for which \(d_m - c_m \neq m\), and \(m\) is the smallest integer; and let \(s = d_m - c_m\).

Consider \((c_i, d_i)\): if \(s < m\), then \(d_i - c_i = s\) since by assumption \(m\) is the smallest.

Therefore, \(d_m - c_m = d_s - c_s = s\),
\[
or\quad d_m - d_s = c_m - c_s
\]
Let \(z := d_m - d_s = c_m - c_s\). Then there is a diagonal move that transforms one winning \((c_m, d_m)\) point into another winning point \((c_i, d_i)\), which contradicts with the definition of a winning point. Therefore, \(s \leq m-1\) is impossible.

Let’s now assume that \(s \geq m+1\).

Observe that \(d_m - 1 \neq d_m\), otherwise for \(k \leq m-1\)
\[
c_i \geq c_m
\]
Consider Hugo’s move \((c_m, d_m - 1)\), where \(d_m - 1 - c_m \geq m\). But in this case Cora cannot make either a horizontal move or a diagonal move that transforms the Game into a winning state. The latter implies that \((c_m, d_m - 1)\) is a winning state, which in its turn contradicts the earlier assumption that \((c_m, d_m)\) is the winning state. Q.E.D.

**P7. Theorem 1**: \{Fundamental property of the winning points\}:

a) Let \(1 \leq x_k \leq 2\); and for all \(k \geq 0\)
\[
x_{k+1} := \left( x_k^2 + 1 \right) / \left( 2x_k - 1 \right)
\]
b) Let \(G = \lim x_k\);  
c) \((S, L)\) is a winning point if
\[
S = \left\lfloor (L - S)G \right\rfloor
\]
For the sake of simplicity of further discussion, we assume that in every point \((c,d)\ c<d\).

4. Game in Progress: An Example

Let \((S_0,L_0) = (29,51)\) be a randomly generated initial point; Hugo makes the 1st move; in italics are shown Hugo’s moves; in bold are Cora’s winning points Table 1.

5. System of Equations with Infinite Sequences

**Proposition 3:** 1). Let \(A = \{ a_n \}; B = \{ b_n \}\) be monotone increasing sequences of positive integers; here \(a_1 = 1\) and \(n = 1,2,\ldots\);

2). Let the sequences \(A\) and \(B\) satisfy the following system of equations:

\[
B \cap A = \emptyset \quad \text{(6)}
\]

\[
B - A = N \quad \text{(7)}
\]

\[
B \cup A = N \quad \text{(8)}
\]

where in (7) \(B - A = \{ b_n - a_n \}\), i.e. \(B - A\) is a sequence of pair-wise differences of corresponding elements of \(B\) and \(A\), and \(N\) is the set of all natural numbers \(\{1,2,3,\ldots\}\). Then the system of Equations (6)-(8) with unknown sequences \(A\) and \(B\) has a solution.

**Proof** [by induction]: The following algorithm is a constructive proof that a solution of (6)-(8) exists. Indeed, the sequences \(A = \{ a_n \}; B = \{ b_n \}\) can be iteratively generated using an analogue of the Sieve of Eratosthenes:

**StepL1:** \((a_1,b_1) = (1,2)\);

**StepL2:** Let \(A_{k-1} = \{ a_1, a_2, \ldots, a_{k-1} \}\); \(B_{k-1} = \{ b_1, b_2, \ldots, b_{k-1} \}\) be sequences such that for every \(k<n\) the following conditions hold: \(a_1 < a_2 < \ldots < a_{k-1}\);

\[
b_1 < b_2 < \ldots < b_{k-1} \quad \text{(9)}
\]

\[
A_{k-1} \cap B_{k-1} = \emptyset \quad \text{(10)}
\]

and for every \(1 \leq i \leq k-1\)

\[
b_i - a_i = i \quad \text{(11)}
\]

**StepL3:** Let \(J = \{ j : b_j \geq a_{n+1} + 1 \}\);

**StepL4:** Compute an integer \(u := \min x \), where \(x > a_{n+1}\) and for all \(i \in J \ x \neq b_i\);

**StepL5:** \(a_n := u\);

**StepL6:** \(b_n := a_n + n\).

Then the conditions (9)-(11) also hold for \(k = n\). Q.E.D.

Applying the Steps L1-L6, we sequentially generate the winning points:

\[
W = \{ (1,2); (3,5); (4,7); (6,10); (8,13); (9,15); (11,18); (12,20); (14,23); (16,26); (17,28); (19,31); (21,34); (22,36); (24,39); (25,41); (27,44); \ldots \}
\]

i.e., \((a_{18},b_{18}) = (27,44)\).

Therefore from the StepL3 \(J = \{11,12,13,14,15,16,17\}\), and \(u = 29\).

Then \((a_{18},b_{18}) = (29,47)\).

6. Alternative Formulation of L1-L6 Algorithm

**Theorem 2:** For every integer \(n \geq 1\)

\[
w_n := (a_n,b_n) \quad \text{(12)}
\]

**Proof**: The following sequence of steps is a constructive proof of Theorem 1. Indeed, let

\[
m := L - S \quad \text{(13)}
\]

**T1.** If \(S = a_m\), then by (3), (7) and (13) \(b_m = L\); [Hugo is now in the winning point].

**T2.** If \(S > a_m\), then Cora selects \(y := S - a_m\); \(S := S - y\) and \(L := L - y\); since \(S > a_m\) implies that \(L > b_m\). Indeed \(L = S + m > a_m + m = b_m\).

**T3.** If \(S < a_m\), then Cora finds either an index \(k < m\) such that \(a_k = S\) or an index \(i < m\) such that \(b_i \subseteq S\).  

**T3a.** If there exists an integer \(k < m\) such that \(a_k = S\), then we select \(L := b_k\); [both \(S < a_m\) and \(a_k = S\) imply that \(k < m\) and \(L > b_k\). Indeed, an assumption that \(k \geq m\) leads to a contradiction, because \(m \leq k\) implies that \(S < a_m \leq a_k\), but \(a_k = S\)]

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player</td>
</tr>
<tr>
<td>Hugo</td>
</tr>
<tr>
<td>Cora</td>
</tr>
<tr>
<td>Hugo</td>
</tr>
<tr>
<td>Cora</td>
</tr>
<tr>
<td>Hugo</td>
</tr>
<tr>
<td>Cora</td>
</tr>
<tr>
<td>Examples of moves</td>
</tr>
<tr>
<td>Type of move</td>
</tr>
</tbody>
</table>

Copyright © 2010 SciRes.
8. Iterative Algorithm and its Complexity

In applications for computer games, an iterative computation of \( a_n \) and \( b_n \) for a large \( n \) is time consuming, since its time complexity \( T(n) \) and space complexities \( S(n) \) are both of order \( O(n) \). For instance, if \( n = 10^{12} \), then we need to generate and store one trillion pairs of integers. A brief analysis shows that this is well beyond of current size of memory for PC. A more efficient algorithm is described below.

9. Direct Computation of \( a_n \) and \( b_n \)

To decrease the complexity of computation of \( a_n \) and \( b_n \) and avoid excessive storage, let’s find a closed-form expression for \( a_n := v(n) \). Then from (11)

\[
b_n := v(n) + n \tag{14}
\]

**Conjecture 1**: (properties of winning points):

C1. \( a_n / m = z + o(m) \);

and \( \lim_{m \to \infty} a_n / m = z \) \( \tag{15} \)

where \( z \) is a constant.

C2. For every integer \( n \geq 1 \)

\[
a_n = \lfloor nz \rfloor \tag{16}
\]

The property (16) and the asymptotic behavior (15) are observed in numerous computer experiments.

**Conjecture 2**: For large \( n \)

\[
b_n / a_n = z + o(n) \tag{17}
\]

**Remark 2**: The Conjecture 2 is also based on extensive computer experiments.

**Theorem 3**: Conjecture 1 implies that \( z \) is an irrational number.

**Proof**: An assumption that \( z \) is a rational number leads to contradiction. Assume that \( z = \frac{q}{s} \), where both \( q \) and \( s \) are relatively prime integers. Then there exists an infinite number of pairs \( a_n \) and \( b_n \) such that \( a_n = b_n \). Indeed, select

\[
n := (q + s)st \quad \text{and} \quad r := qst \tag{18}
\]

Then for the integer \( t = 1, 2, 3, \ldots \) it follows from (16) and (14) that

\[
a_n = (q + s)qt \tag{19}
\]

and \( b_n = q^2 t + qst \), \( \tag{20} \)

which is a violation of the conditions (6) and (10).

**Conjecture 3**: \( z = g + 1 \), where \( g \) is a golden ratio, i.e.,

\[
g = (\sqrt{5} - 1)/2 \tag{21}
\]

The property (21) is observed in numerous computer experiments. It plausibility follows from the following:

Since for a large \( n \)

\[
a_n = nz + o(n) \quad \text{and} \quad b_n = (z+1)n + o(n) \tag{22}
\]

then it follows from (17) and (22) that

\[
b_n / a_n \approx (z+1)/z \approx z \tag{23}
\]

Then for large \( n \)'s, \( z \) is a positive solution of the equation.

\[
x^2 - x - 1 = 0 \tag{24}
\]

i.e., \( z \approx (\sqrt{5} + 1)/2 \), \{the value of the golden ratio \( +1 \} \}

Let the Game be in the state \( (S, L) \) after Hugo’s move and let

\[
m := L - S (13) \]

Then the Game is implicitly in one of five states \{where by convention holds that \( S < L \}:

A. \( (S, L) = (a_m, b_m) \), \{the game is in a winning state for Hugo\};

B. \( (S, L) = (a_i, a_j) \), where \( i < j \);

C. \( (S, L) = (a_i, b_j) \); where either \( i < j \) or \( i > j \);

D. \( (S, L) = (b_i, a_i) \), \( i < j \);

E. \( (S, L) = (b_i, b_j) \), \( i < j \).

However, from the condition (11) alone we do not know yet in which of the states \( A, B, C, D \) or \( E \) the Game is.

10. Algorithm for Winning Points (AWP)

\( A1: \) Let \( m := L - S \);

\( A2: \) Using (16) and (21), compute \( a_m \);

if \( a_m = S \) then by (11) \( b_m = L \); \{Hugo is now in the winning state \( w_m \} \}

\( A3: \) if \( S > a_m \) then do \( y := S - a_m \); \( S := S - y \) and \( L := L - y \);

\( A4: \) if there exists an integer \( k < m \) such that \( a_k = S \) then \( L := b_k \);

else find an integer \( i < m \) such that \( b_i = S \); \( L := S \); \( S := a_i \).

11. Validation of AWP

V1. In A3, \( S > a_m \) implies that \( L > b_m \).

Then \( L = S + m > a_m + m = b_m \);

V2. If \( S < a_m \), then there exists either an integer \( k < m \) such that \( a_k = S \) or an integer \( i < m \) such that \( b_i = S \);

V3. Both \( S < a_m \) and \( a_k = S \) imply that \( k < m \) and \( L > b_k \). Indeed, an assumption that \( k \geq m \) leads to a contradiction, because \( m \leq k \) implies that \( S < a_m \leq a_k \), but \( a_k = S \);
V4. In A4, \( b_i = S \) implies that \( a_i < L \). Hence \( a_i < b_i = S < L \).

**Example 2** \((\text{case } S > a_n)\): Let the Game be in the state \((S, L) = (19, 26)\) after the Hugo’s move.

Because \( m = 26 - 19 = 7 \), compute \( a_7 = \lceil 7(g + 1) \rceil = 11 \) and \( b_7 = 18 \).

Since \( 11 < 19 \) and \( 18 < 26 \), then Hugo moves \( S = S - m = S - 7 \) and \( L := L - m = L - 7 \);

**Example 3** \((\text{case } S < a_n)\): Let now after Hugo’s move \((S, L) = (15, 32)\).

Since \( m = 32 - 15 = 17 \), compute \( a_{17} = \lceil 17(g + 1) \rceil = 27 \).

Since \( 15 < a_{7} \), but \( 15 = b_7 \), then Cora’s move is \( L := L + 15 \) and \( S := S - 15 \).

**Example 4** \((\text{case } S < a_n)\): Let \((S, L) = (14, 29)\) after Hugo’s move.

Since \( m = 29 - 14 = 15 \), compute \( a_{15} = \lceil 15(g + 1) \rceil = 24 \).

Since \( a_{15} > 14 \), but \( 14 = a_{9} \), then Hugo moves \( L := L - 14 = 26 \).

**Example 5** \((\text{case } a_n = S)\): Then \( b_n = L \), and the game is in the winning state for Hugo.

### 12. Fibonacci Properties of Winning Points

1. If \( n \) is an odd Fibonacci number, \( i.e., \) if \( n = F_{2k-1} \),
   then
   \[
   a_n = F_{2k} \tag{25}
   \]

2. If \( n \) is an even Fibonacci number, \( i.e., \) if \( n = F_{2k} \),
   then
   \[
   a_n = F_{2k+1} - 1 \tag{26}
   \]

Indeed, \( a_{F_1} = 3; a_{F_2} = 8; a_{F_3} = 21; a_{F_4} = 55 \); 
But \( a_{F_2} = 1; a_{F_3} = 4; a_{F_4} = 12; a_{F_5} = 33 \).

### 13. Solution of Equation with Unknown Index

On the step A4 of the algorithm we must solve either equation \( a_k = S \) or \( b_i = S \) in order to respectively determine the indices \( k \) or \( i \). In order to determine the indices we must solve either the equation

\[
a_k = S \tag{27}
\]

or the equation

\[
b_i = S \tag{28}
\]

This can be done by using (16)

\[
a_k = \lfloor k(x + 1) \rfloor = S \tag{29}
\]

1) Find the smallest integer \( k^* \) satisfying the inequality \( k > S \); if \( \{k^{+} \} \leq S \) then

\[
k = k^* ; \quad a_{k^*} = S \tag{30}
\]

2) If \( i^* \) is the smallest integer satisfying the inequality \( i(g+1) > S \)

then

\[
i = i^* \quad \text{and} \quad b_{i^*} = S \tag{31}
\]

If (16) has an integer solution, then from (29) we find the smallest integer \( k^* \) satisfying the inequality

\[
k^* (g+1) > S \tag{32}
\]

Otherwise, \( \{k(g+1) \} \neq S \) we solve the equation

\[
b_i = S \tag{33}
\]

**Example 6:** Find an integer index \( k \) such that \( a_k = 102 \). Then \( k^* = 64 \) is the smallest integer for which holds

\[
k^* \leq 102/(g+1) \tag{see (29)}.\]

### 14. Required Accuracy for \( g \)

It is assumed that in the Examples 3-6 and 8 we know the exact value of an irrational number \( g \). However, to find an integer solution of (17) for an arbitrary large index \( k \) or \( i \) we must compute \( g \) with a high precision. Let

\[
g = d_1 / 10 + d_2 / 10^2 + \ldots + d_r / 10^r + \ldots \tag{34}
\]

where \( d_i \) is the \( i \)-th decimal digit of \( g \) and

\[
g(t) = \left\lfloor 10^t g \right\rfloor / 10^t =
\]

\[
d_1 / 10 + d_2 / 10^2 + \ldots + d_r / 10^r \tag{35}
\]

i.e., \( g(t) \) contains only the first \( t \) decimal digits of \( g \).

**Theorem 4:** Let \( n \leq 10^k \), \( a_n = S \)

Then for all \( t \geq k \) also holds that

\[
d_n^{(i)} = \left\lfloor n g(t) \right\rfloor = S. \tag{37}
\]

### 15. \( O(\log\log n) \) Time Complexity for Winning Strategies

It is easy to verify that a positive root of (24) can be computed using a Newton iterative process

\[
x_{r+1} := \left( x_r^2 + 1 \right)/(2x_r - 1),
\]

Where

\[
x_0 := 1.618 \tag{38}
\]

The process (38) has the following properties:

a). It converges to \((1 + \sqrt{5})/2\), i.e., for large \( r \)

\[
x_r = (1 + \sqrt{5})/2 + \varepsilon_r, \tag{39}
\]

where \( \varepsilon_r \) is a degree of accuracy (error) after \( r \) iterations.
b). The error \( \varepsilon_r \) satisfies the inequality
\[
\varepsilon_r \leq \varepsilon_0 = |x_0 - g|^2,
\]
i.e., it has a quadratic rate of convergence, and
\[
\varepsilon_0 < 0.001 = 10^{-3}, \quad (40)
\]
Then from the inequality \( 10^{3k_2} \leq 10^{-k} \) we derive that
\[
3 \times 2^r \geq k \quad (41)
\]
Thus
\[
r \geq \left\lceil \log_2 \left( \frac{k}{3} \right) \right\rceil \geq \left\lceil \log_2 \left( \left\lfloor \log_{10} n \right\rfloor / 3 \right) \right\rceil \approx \log_{10} \log_{10} n
\]
(42)
The inequalities (42) are derived from (36), (40) and (41). Then from analysis of (37) it follows that the time complexity
\[
T(n) = O(\log \log n)
\]
The Table 2 shows how many Newton iterations \( r(n) \) are required to compute \( a_n \) as a function of \( n \).
In addition, we do not need to store any winning points. Instead, as it is demonstrated below, only a single real value of \( g \) must be stored.
However, \( \left\lfloor 64(g+1) \right\rfloor = 103 \neq 102 \). Hence the equation \( a_1 = 102 \) does not have a solution. On the other hand, \( b_1 = 102 \) does have a solution. Indeed, from (33) it follows that \( i \geq 102/(g+2) = 38.961 \), i.e., \( i^* = 39 \). And finally \( \left\lfloor 39(g+2) \right\rfloor = 102 \).

### 16. Solution of Index Equations Revisited

**R0.1.** Let \( S := s; L := l \); where both integers \( s, l \) are generated randomly at the beginning of the Game; let \( t := L - S \);

**R0.2.** \( r := \left\lfloor \log t \right\rfloor \); using the iterative process (36), compute
\[
x_r; \quad (43)
\]

**R0.3.** Let
\[
g_r := x_r; \quad (44)
\]
{during the entire Game use \( g_r \) as an approximation of \( g \) in the Equations (29) or (33)};

**R1.** Find the smallest integer \( k^* \) satisfying the inequality
\[
k g_r > S \quad (45)
\]
if \( \left\lfloor k^* g_r \right\rfloor = S \)
then \( k = k^*; \ a^*_r = S; \quad (46) \)

**R2.** If \( i^* \) is the smallest integer satisfying the inequality
\[
i^* \ (g_r + 1) > S \quad (47)
\]

<table>
<thead>
<tr>
<th>( n = 10^i )</th>
<th>[0,3]</th>
<th>[4,6]</th>
<th>[7,12]</th>
<th>[13,24]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r(n) )</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\text{then } i^* = i \text{ and } b_r = S \quad (48)
\]

**Example 7:** Let at the beginning of the Game \( s := 2,718,282 \) and \( l := 3,141,593 \).
Then \( m := L - S = l - s = 323,311 < 10^6 \). From the inequality \( 6 \geq 3 \times 2^r \), {see (40) and (41)}, it follows that \( r = 1 \).
Hence, only one iteration of (38) is necessary to find \( g_r \) with required accuracy.

### 17. The Algorithm

It is assumed that \( Hugo \) makes the first move by randomly generating positive integers \( S_0 \) and \( L_0 \) such that
\[
L_0 \geq (e-1)S_0 \geq Q \quad (49)
\]
where \( e \) is Euler number, {see Remark3 below};
\( V: m := L - S; \)

if \( m = 0 \) then \( z := S; S := S - z; L := L - z; \{ \text{end of the Game: Cora is the winner}\}; \)
else
\[
t := \left\lfloor \log_2 \left( t / 3 \right) \right\rfloor; \quad x_0 := 1.618;
\]
for \( k \) from 0 to \( r - 1 \)
do
\[
x_{k+1} := (x_k^2 + 1) / (2x_k - 1); \quad G := x_r; \quad a_r := \left\lfloor Gm \right\rfloor; \]
if \( S = a_r \) then the Game is already in the winning state for Hugo;
\{Nevertheless Cora might decide to continue the Game hoping that Hugo will make a mistake, i.e., he will “miss the point”\};
if \( L > 3 \) then with \( prob = 1/2 \): \( c := 1 \text{ or } 2; \ L := L - c \}; \)
goto \( V; \)
else if \( S > a_r \) then
\( z := S - a_r; L := L - z; S := S - z; \quad \text{ goto } V; \) else
\[
k := \left\lfloor S / G \right\rfloor; \quad (50)
\]
\[
\left\lfloor kG \right\rfloor = S \text{ then } y := m - k; L := L - y; \quad (51)
\]
else \( i := \left\lfloor S / (G + 1) \right\rfloor; \quad a := \left\lfloor Gi \right\rfloor; \quad u := L - a; \text{ temp := } S; S := L - u; L := \text{ temp }; \quad \text{ goto } V. \)
Remark 3: In order to assure that the first randomly generated point is not a winning point, it is sufficient to select such $S_0$ and $L_0$ that
\begin{equation}
S_0 \neq \left( (L_0 - S_0) (g + 1) \right)
\end{equation}
That is guaranteed by (47) and (15), since
\begin{equation}
L_0 \geq 1.718 S_0 \neq 1.618 S_0 .
\end{equation}

18. Randomization

Let $n_a$ and $n_b$ be the number of integers on interval $[1, M]$ such that $1 \leq a_k \leq M$ and $1 \leq b_k \leq M$ respectively, i.e., $n_a + n_b = M$.

Then
\begin{equation}
n_a(M) = M / (g + 1)
\end{equation}
and
\begin{equation}
n_b(M) = M / (g + 1)^2
\end{equation}

Hence, if a pair of integers $(S, L)$ is generated randomly, then it is more likely that they will be elements of the sequence $A$, than the sequence $B$.

Remark 4: The sequence of the operations (50) and (51) in the Algorithm is based on the observation that for every $M$, $n_a(M) > n_b(M)$.

That is why on the A4 we first check whether there is a solution of $a_k = S$ and only then whether there is a solution of $b_k = S$. This sequence of verifications decreases the average complexity of the algorithm. Another approach is to randomize the sequence of these operators: Namely, with the probability $g = 0.618$ to execute (50) and then, if necessary, to execute (51). And with the probability $g = 0.382$ to execute (51) and only then, if necessary, to execute (50).

Example 8: If $M = 50$, then $n_a(50) = 31$ and $n_b(50) = 19$. Thus, if $u$ is an arbitrary selected integer on the interval $[1, 50]$, then with probability $g$ there exists an index $k$ such that $a_k = u$, and with probability $g^2 = 0.382$ there exists an index $i$ such that $b_i = u$.

19. The First Move

Without a third independent party, it seems impossible to introduce a random and trustworthy mechanism for deciding whose move is the first. As a palliative solution, the following procedure is suggested: immediately, after the first point $(S_0, L_0)$ is generated, Hugo has a short period of time (say, a couple of seconds) to decide who must make the first move. One way to preclude Hugo from cheating and to introduce more variety to the Game, select $Q := 2Q$ on every consecutive run of the Game with the same player. More detailed analysis of possible alternatives is beyond the scope of this paper.

20. Varieties of Nim-Game on Plane

Of many possible varieties I consider only two: the Attrition game and the Flip-Flop game.

In both games the moves are the same as in the Game described above in this paper. Only the goals are different.

Attrition game: The first player that reaches point $(0, 0)$ is a loser.

Flip-flop game: Only once during the Game players on their move can change the goal of the Game if
\begin{equation}
L \geq S + 2 \geq 7
\end{equation}

21. Winning Strategies

Let the winning points $w$ in the Attrition game. It is clear that both $w = (0,1)$ and $w = (2,2)$ are the winning points for Cora. Indeed, after Cora’s move $(0, 1)$ Hugo is losing the Game. The same is with $(2,2)$: after that move Hugo is forced to reach $(0,0)$, because Hugo can make either $(0,2)$ or $(1,1)$ or $(1,2)$ move. Then Cora moves $(0,1)$ and Hugo has no other choice but move $(0,0)$.

Winning points $f_k$ in Flip-Flop game: Although it seems confusing, actually the winning points for the Flip-Flop Game are very simple. It follows from an observation that for all $k \geq 2$
\begin{equation}
f_k = w_k,
\end{equation}
i.e., $w = (3,5)$; $w = (4,7)$; and only $w = (2,2)$ and $w = (1,0)$.

Hence, if the Game is in the attrition phase, then
\begin{equation}
f_k = w_k, \text{ otherwise } f_k = w_k.
\end{equation}

From the (56) winning strategy it follows that “Only once during the Game” — requirement is inessential and it is introduced for a psychological reason only. The Game can be further modified if the flipper must pay for every flip, and the winner gets the “bank”.

After this paper was complete d, the author discovered that the Game has been described in [13,14].

22. Acknowledgements

I appreciate H. Woźniakowski for his suggestion, an anonymous reviewer for several corrections and B. Blake for comments that improved the style of this paper.
23. References