How to solve large scale deterministic games with mean payoff by policy iteration

Vishesh Dhingra
Department of Mathematics
Indian Institute of Technology Delhi
New Delhi, India
vishesh.dhingra@gmail.com

Stéphane Gaubert
INRIA
Domaine de Voluceau
78153 Le Chesnay Cédex, France
Stephane.Gaubert@inria.fr

ABSTRACT

Min-max functions are dynamic programming operators of zero-sum deterministic games with finite state and action spaces. The problem of computing the linear growth rate of the orbits (cycle-time) of a min-max function, which is equivalent to computing the value of a deterministic game with mean payoff, arises in the performance analysis of discrete event systems. We present here an improved version of the policy iteration algorithm given by Gaubert and Gunawardena in 1998 to compute the cycle-time of a min-max functions. The improvement consists of a fast evaluation of the spectral projector which is adapted to the case of large sparse graphs. We present detailed numerical experiments, both on randomly generated instances, and on concrete examples, indicating that the algorithm is experimentally fast.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory; G.4 [**Mathematical software**]: Algorithm design and analysis

General Terms

Algorithms, Performance

Keywords

Policy iteration, repeated games, graph algorithms, max-plus algebra, nonlinear harmonic functions

1. INTRODUCTION

We present an algorithm to solve a combinatorial game with perfect information, which arises in several fields, including performance evaluation of discrete event dynamic systems.

1.1 Description of the game

Let $G = (V, E)$ denote a directed bipartite graph, in which every arc $(i, j) \in E$ has a weight, $r_{ij} \in \mathbb{R}$. We assume that every node is the tail of at least one arc. Two players, "Max", and "Min", move a pawn on this graph, according to the following rules. The pawn is initially at a given node $i_0 \in V$. The player who plays first, say Max, chooses an arc (i_0, i_1) in E, moves the pawn from i_0 to i_1, and Min pays $r_{i_0 i_1}$ to him. Then, Min chooses an arc (i_1, i_2) in E, moves the pawn from i_1 to i_2, and pays $r_{i_1 i_2}$ to Max. The game continues in this way, alternating the moves of Max and Min. The reward of Max (or the loss of Min) after k turns, if the trajectory is $(i_0, i_1, \ldots, i_{2k})$, is given by $r_{i_0 i_1} + \cdots + r_{i_{2k-1} i_{2k}}$. We are interested in the reward (or loss) and optimal strategies of the two players when the horizon of the game (i.e., the number of turns) is large.

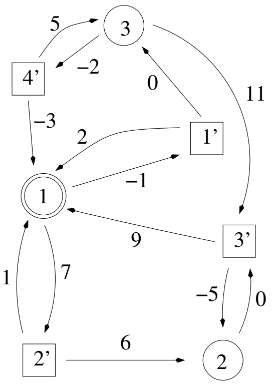


Figure 1: Zero-sum deterministic game

To get the intuition of this problem, consider the example of Fig. 1. The circles (resp. squares) represent the nodes at which Max (resp. Min) can play. The bipartite nature of the graph reflects the fact that the two players alternate their moves. The initial node, "1", is indicated by a double circle. So, Max can initially move the pawn either to node $1'$, with reward -1, or to node $2'$, with reward 7. Assume first that Max moves (greedily) the pawn to node $2'$, receiving 7. Then, Min may move the pawn to node 2, paying 6 to Max. Now, Max must move the pawn to node $3'$, receiving 0, and Min can move the pawn back to node 2, receiving 5 from Max. Hence, Min has a strategy ensuring her an average win per turn of 5 if Max's initial move is from 1 to $2'$.

Assume now that Max's initial move is from 1 to $1'$, Then, the circular sequence of moves $1 \xrightarrow{-1} 1' \xrightarrow{0} 3 \xrightarrow{-2} 4' \xrightarrow{-3} 1$ ensures to Min an average win per turn of $(1+0+2+3)/2 = 3$. Max has no better option than moving from 3 to $4'$, because moving from 3 to $3'$ would lead him again to the circuit $3' \to 2 \to 3'$, in which he looses 5 per turn, instead of 3. It follows that Max minimal mean loss per turn is of 3.

The algorithm described in this paper allows one to solve such games, for very large graphs. In the remaining part of the introduction, we give additional motivation, present the main technical ingredients of the algorithm, and give references.

1.2 Cycle-time of min-max functions and limit of the finite horizon game

We partition the set of nodes as $V=\{1,\dots,p\}\cup\{1',\dots,n'\}$, where $1,\dots,p$ denote the nodes in which it is Max's turn to play, and $1',\dots,n'$ denote the nodes in which it is Min's turn to play. Since both players alternate their moves, the arcs of G are of the form (j',i) or (i,j') with $1 \le j \le n$ and $1 \le i \le p$. We define the dynamic programming operators

$$g : \mathbb{R}^p \to \mathbb{R}^n,\ g_j(y) = \min_{(j',i)\in E} r_{j'i} + y_i, \quad \forall y \in \mathbb{R}^p \qquad (1)$$

and

$$g' : \mathbb{R}^n \to \mathbb{R}^p,\ g'_i(x) = \max_{(i,j')\in E} r_{ij'} + x_j, \quad \forall x \in \mathbb{R}^n \ . \qquad (2)$$

We set:

$$f = g \circ g' \ .$$

Maps of this form are called *min-max functions* after the work of Olsder [23] and Gunawardena [14].

For instance, the maps corresponding to the game of Fig. 1 are given by:

$$g(y) = \begin{pmatrix} (2+y_1)\wedge y_3 \\ (1+y_1)\wedge(6+y_2) \\ (9+y_1)\wedge(-5+y_2) \\ (-3+y_1)\wedge(5+y_3) \end{pmatrix},$$

$$g'(x) = \begin{pmatrix} (-1+x_1)\vee(7+x_2) \\ x_3 \\ (-2+x_4)\vee(11+x_3) \end{pmatrix}.$$

Here, $\vee$ and $\wedge$ denote respectively the max and min operations.

In the sequel, it will be more convenient to analyse the case in which Min plays first. The other case reduces to it by changing the sign of payments and exchanging the roles of players.

Consider the version of the game in which the horizon is N, and the initial state is j', for some $1 \le j \le n$. In this context, we call (feedback) *strategy* a decision rule of one player, depending only on the current state, and on the number of turns remaining to play. For all pairs of strategies (σ',σ) of Max and Min, let $R_{j,N}(\sigma',\sigma)$ denote the corresponding reward of Max after N turns. The *value* of the game in horizon N, v_j^N, is defined as the unique scalar γ such that Max has a strategy $\bar\sigma'$ ensuring him a win of at least γ, for all strategies σ chosen by Min, and Min has a strategy $\bar\sigma$ ensuring her a loss of at most γ, for all strategies σ' of Max. So

$$R_{j,N}(\sigma',\bar\sigma) \le v_j^N \le R_{j,N}(\bar\sigma',\sigma), \qquad \forall \sigma',\sigma \ .$$

Standard dynamic programming arguments show that the *value function* of Max:

$$v^N := (v_j^N)_{1\le j\le n} \in \mathbb{R}^n$$

satisfies

$$v^N = f(v^{N-1}), \qquad v^0 = 0 \ .$$

Define the *cycle time* of f to be the quantity

$$\chi(f) := \lim_{k\to\infty} f^k(0)/k \ ,$$

where f^k denotes the k-th iterate of f. Thus, $\chi_j(f)$ is the mean reward per time unit as the horizon tends to infinity, if the initial state is j'.

The problem which motivates this paper is the following:

Problem 1. *Given a min-max function f, compute $\chi(f)$.*

The term *cycle time* originates from the discrete event systems literature, where dynamical systems of the form $x(k) = f(x(k-1))$ occur: the entries of $x(k)$ represent the times of k-th occurrence of repetitive events. Hence, the i-th coordinate $\chi_i(f)$ represents the average time between consecutive occurrences of the event with label i. The problem of the existence of the cycle time of min-max functions arose after the work of Gunawardena [14], motivated an earlier work of Burns [4] concerning the performance analysis of asynchronous digital circuits. Some problems of scheduling with no wait, and some problems of feedback synthesis for discrete event systems [20] can be reduced to max-plus linear equations, which reduce to fixed point problems for min-max functions, see Section 4.3.

1.3 Invariant half-lines

Before showing how Problem 1 can be solved, we need the notion of invariant half-line. Kohlberg [21] showed that any map f from $\mathbb{R}^n$ to $\mathbb{R}^n$ which is piecewise affine and nonexpansive in some norm admits an *invariant half-line* (on which it acts by translation), meaning that there exist vectors $v, \eta \in \mathbb{R}^n$ such that

$$f(v + t\eta) = v + (t+1)\eta, \qquad (3)$$

for all scalars t large enough.

This implies in particular that if f is a min-max function, then, $\chi(f)$ exists and

$$\chi(f) = \eta \ .$$

Indeed, it can be checked that min-max functions are nonexpansive in the sup-norm, meaning that $\|f(x) - f(y)\|_\infty \le \|x - y\|_\infty$. Hence, the existence and the value of the limit $\lim_{k\to\infty} f^k(x)/k$ are independent of the choice of $x \in \mathbb{R}^n$. Taking $x = v + t\eta$, with t sufficiently large, we deduce that $\chi(f) = \eta$. In this paper, we solve the following problem, which is more precise than Problem 1.

Problem 2. *Compute an invariant half-line of a given min-max function.*

Note that in discrete event systems problems, the invariant half-line is of independent interest: it can be interpreted as a stationary schedule.

1.4 Policy iteration algorithm for deterministic games

The proof of Kohlberg [21], or the proof of the more general results of Bewley and Kohlberg [2], rely in essence on quantifier elimination arguments. Different methods must be developed to obtain practicable algorithms.

Gaubert and Gunawardena [13], extending an earlier work of Cochet and the same authors [8] in a special case, proposed an algorithm to compute $\chi(f)$. This algorithm uses the idea of *policy iteration*, i.e., of iteration in the space of strategies of one player.

Assume first that Min fixes a feedback stationary strategy σ, meaning that her decision depends only on the current state and is independent of the time. So, the strategy σ can be identified as a map from $\{1',\dots,n'\}$ to $\{1,\dots,p\}$, that we still denote by σ, assigning to each of the nodes $1',\dots,n'$ one node $\sigma(i')$ such that $(i',\sigma(i'))\in E$. Once Min's strategy is known, Max must solve a one player game, the dynamic programming operator of which, denoted by $f^{(\sigma)}$, is given by:

$$f_i^{(\sigma)}(x) = r_{i'\sigma(i')} + \max_{(\sigma(i'),j')\in E} r_{\sigma(i'),j'} + x_j \ . \qquad (4)$$

Maps of this form are called *max-plus linear maps.* Computing an invariant half-line of a max-plus linear map is a much simpler problem [6], which essentially reduces to computing the maximal circuit mean in a digraph.

In a nutshell, the policy iteration algorithm of [13] constructs a sequence of strategies and associated invariant half-lines with the property that at every step, the cycle time $\chi(f^{(\sigma)})$ and $\chi(f^{(\pi)})$ of the old and new strategies σ and π satisfy:

$$\chi(f^{(\pi)}) \le \chi(f^{(\sigma)}) \ . \qquad (5)$$

The algorithm stops when the invariant half-line of the current strategy is an invariant half-line of f. If the inequalities (5) are always strict, the same strategy cannot be selected twice, and since the total number of (feedback, stationary) strategies is finite, the algorithm must terminate.

The difficulty stems from *degenerate iterations*, in which the equality holds in (5). Naive implementations of policy iteration, in which degenerate iterations are not properly handled, can lead the algorithm to cycle.

This difficulty is solved in [8, 13], where it is shown that there exists a canonical choice of an invariant half-line of the map f^{σ}, which guarantees that the algorithm terminates. This choice relies on max-plus spectral theory: at each degenerate iteration, the invariant half-line of $f^{(\pi)}$ is chosen to be the image of the invariant half-line of $f^{(\sigma)}$ by the *spectral projector* of $f^{(\pi)}$. The convergence proof relies on a discrete max-plus analogue of the classical result of potential theory, showing that a function which is harmonic in the usual sense on a regular bounded domain of $\mathbb{R}^n$ is defined uniquely by its trace on the boundary. The max-plus analogue of the "boundary" is defined in terms of the nodes belonging to circuits of maximal weight-to-length ratio (critical nodes).

One iteration of the algorithm of [8] takes a time $O(nm)$, where m is the number of arcs of G, plus the time needed to apply a spectral projector to a given super-harmonic vector if the iteration is degenerate. An explicit formula for the spectral projector, due to Cohen, Dubois, Quadrat, and Viot (see [9, 8]), is known. It would lead to an execution time of $O(n^3)$ for every iteration of the algorithms of [8, 13], and it requires a space $O(n^2)$. In this paper, we present a detailed and improved implementation of the algorithm of [13]. Here, the image of a given super-harmonic vector by the spectral projector is computed in $O(nm)$ time and $O(n)$ space. This allows us to solve large games. The proofs, which use the ideas of [8, 13], will appear elsewhere.

We present detailed experiments, both on random and concrete examples, showing that the algorithm is experimentally fast.

For random sparse graphs, the number of iterations (number of strategies of the minimiser) turns out to grow sublinearly with the dimension. In certain structured examples, however, the number of iterations can be of order n. We give the example of a pursuit-evasion game, in a room with an obstacle, in which the number of iterations is of the order of the diameter of the room.

It is an open question to determine whether or not the algorithm runs in polynomial time. It is in fact an important open problem to show that $\chi(f)$ can be computed in polynomial time. There is an indication that the answer to the latter problem may be positive, because Condon [10] showed that decision problems for games of this nature are in the complexity class NP $\cap$ co-NP. Note that $\chi(f)$ can be computed in pseudo-polynomial time, as shown by Zwick and Paterson [25].

1.5 Related works

To conclude this introduction, let us make additional comments on the history of the problem, and mention other references.

The problem of the existence of the cycle time is intimately related with the problem of the existence of the value for the associated infinite deterministic game with mean payoff, in which the payment associated to an infinite trajectory is the limsup of $(r_{i_0 i_1}+\dots+r_{i_{2k-1}i_{2k}})/k$ as k tends to infinity. The value was shown to exist by Ehrenfeucht and Mycielski [12]. In [15], Gurvich, Karzanov, and Khachiyan gave an algorithm to compute the value of such games. They mention the existence of a class of graphs for which the number of iterations of their algorithm can be exponential, although they report that the number of iterations grows linearly with the dimension for random examples.

Another line of research concerns policy iteration. The technique of policy iteration was invented by Howard [17], to solve stochastic control problems. Hoffman and Karp [16] generalised the idea to the case of stochastic games. Their generalisation requires transition probabilities to be positive. Hence it does not apply to the case of deterministic games. To our knowledge, the first application of policy iteration techniques to the latter case appeared in [8, 13]. Another policy iteration algorithm, inspired by the ones of [8, 13] has been proposed by Cheng and Zheng [5]: their algorithm differs from the one of [8] in the way that it computes inductively the different coordinates of the cycle time, starting from the largest one. Policy iteration techniques for "parity games" (which reduce to a subclass of mean payoff games) have been developed with motivations from Model checking, see Vöge and Jurdziński [24]. See also Costan et al. [11] for an application of policy iteration techniques to the static analysis of programs. Finally, we note that the question of the complexity of policy iteration for games has recently received a considerable attention, see Björklund, Sandberg, and Vorobyov [3] and Jurdziński, Paterson, and Zwick [18]. No polynomial time algorithm is currently known.

2. COMPUTING MAX-PLUS SPECTRAL PROJECTORS

In this section, we present the max-plus analogues of some results of potential theory, which will be needed to give an improved implementation of the algorithm of [13].

2.1 Spectral projectors of max-plus linear operators with zero cycle time

Let $A = (A_{ij}) \in (\mathbb{R} \cup \{-\infty\})^{n\times n}$ denote a matrix with at least one finite entry in every row. The *max-plus linear operator* associated to A, also denoted by A, is the map from $\mathbb{R}^n$ to $\mathbb{R}^n$ defined by:

$$x \mapsto Ax, \quad (Ax)_i := \max_{1\leq j\leq n} A_{ij} + x_j \enspace .$$

In this section, we assume that $\chi(A)$ is zero, and show that the image of a super-harmonic vector by the spectral projector of A can be computed in time $O(nm)$ and space $O(n)$. This result will be extended to the case of a non-zero cycle time in Section 2.2.

We denote by concatenation the composition of max-plus linear operators. For all $I, J \subset \{1, \dots, n\}$, we denote by A_{IJ} the $I \times J$ submatrix of A. For all vectors $x \in \mathbb{R}^n$, we denote by x_I the vector of $\mathbb{R}^I$ obtained by restricting the map $i \mapsto x_i$ to I. We denote by $\vee$ the pointwise supremum of vectors or matrices. Recall that the *Kleene star* of a max-plus linear operator A is given by $A^* := I \vee A \vee A^2 \vee \cdots$, where I denotes the max-plus identity matrix (with 0 on the diagonal and $-\infty$ elsewhere), and where for all $k \geq 0$, A^k denotes the k-th iterate of the max-plus linear operator A. If $\chi(A) \leq 0$, it is known [1, Th. 3.20] that $A^* = I \vee A \vee A^2 \vee \cdots \vee A^{n-1}$.

We shall need the characterisation of the cycle-time of a max-plus linear map.

Recall that to the matrix A is associated the *digraph* $G(A)$, with nodes $1, \dots, n$, and with an arc from i to j with weight A_{ij} if $A_{ij} \neq -\infty$. (We warn the reader that the reverse orientation of arcs is sometimes chosen in the literature.) The *weight* of a path is defined as the sum of the weights of its arcs. We say that a node *has access* to a circuit in a digraph if there is a path from this node to a node of this circuit. The *weight-to-length ratio* of a circuit $i_1 \to \cdots \to i_k \to i_1$ of the digraph of A is defined by $(A_{i_1 i_2} + \cdots + A_{i_k i_1})/k$. We denote by $\rho_{\max}(A)$ the maximum of the weight-to-length ratios of the circuits of the digraph of A.

Proposition 1 ([8, Prop. 1.3]). *Let $A \in (\mathbb{R} \cup \{-\infty\})^{n\times n}$ denote a matrix with at least one finite entry per row. Then, $\chi_i(A)$ is the maximum of the weight-to-length ratios of all the circuits of $G(A)$ to which i has access.* □

When $\chi(A) = 0$, we define the *critical graph* of A to be the union of the set of circuits with weight 0 of $G(A)$, which are called the *critical circuits*, and we define the set $C(A)$ of *critical nodes* to be the set of nodes of the critical circuits. We say that a vector u is *super-harmonic*, in the max-plus sense, with respect to A, if $Au \leq u$. We say that it is *harmonic* if $Au = u$. The following key result, which is a slightly more precise version of results in [8, Lemma 1.4] and [13, Lemma 1], shows that there is a unique harmonic vector which coincides with a given super-harmonic vector on the set of critical nodes. The final part of this lemma appeared in [7, Lemma 7], it is related to [5, Lemma 2].

Proposition 2. *Let $A \in (\mathbb{R} \cup \{-\infty\})^{n\times n}$ denote a matrix with at least one finite entry per row. Assume that $\chi(A) = 0$. Let $u \in \mathbb{R}^n$ denote a vector such that $Au \leq u$. Let $C := C(A)$ denote the set of critical nodes of A. Then, there is a unique vector v such that $Av = v$ and $v_C = u_C$. It satisfies $v \leq u$, and it is determined by $v_N = A^*_{NN} A_{NC} u_C$.*

Definition 1 (Spectral projector acting on super-harmonic vectors). *We denote by P_A, and we call* spectral projector *of the matrix A, the map sending a super-harmonic vector u to the harmonic vector v defined in Proposition 2.*

Proposition 2 admits the following algorithmic translation.

Algorithm 1 *(Spectral Projector). Input*: A matrix $A \in (\mathbb{R} \cup \{-\infty\})^{n\times n}$, with at least one finite entry per row, such that $\chi(A) = 0$, and a vector $u \in \mathbb{R}^n$ such that $Au \leq u$. *Output*: The unique vector $v \in \mathbb{R}^n$ such $Av = v$ and $v_C = u_C$, where $C := C(A)$ denotes the set of critical nodes of A.

1. Compute the set of critical nodes C of A, let $N := \{1, \dots, n\} \setminus C$ denote the set of non-critical nodes, and let $|N|$ denote the number of non-critical nodes.

2. Compute the max-plus product $b := A_{NC} u_C$.

3. Compute the vectors $z^{(k)}$ defined inductively by

   $$z^{(0)} := b, \qquad z^{(k)} = A_{NN} z^{(k-1)} \vee b,$$

   until $k = |N| - 1$ or $z^{(k)} = z^{(k-1)}$.

4. Return the vector v such that $v_C = u_C$ and $v_N = z$, which is such that $Av = v$.

Algorithm 1 takes a time $O(nm)$ and a space $O(n)$, where m is the number of arcs of $G(A)$. Its correctness follows from Proposition 2, and from the correctness of the Ford-Bellman algorithm. Note that computing $A^*_{NN} A_{NC} b$ is a "all sources-single destination" shortest path problem, which could be solved by other shortest path algorithms (instead of the Ford-Bellman algorithm appearing in the third step).

Example 1. We first illustrate Algorithm 1 by a very simple example. A more complex illustration will appear in Example 2 of Section 2.2. Consider the matrix

$$A = \begin{pmatrix} -\epsilon & -1 \\ -5 & 0 \end{pmatrix} \enspace ,$$

with $2 > \epsilon > 0$. and take $u = (4, 2)^T$. Since $Au = ((-\epsilon + 4) \vee 1, 2)^T \leq u$, u is super-harmonic. The critical graph of A consists of the circuit $(2, 2)$, so $C = \{2\}$ and $N = \{1\}$. We have $b = A_{NC} u_C = -1 + 2 = 1$. Step 3 of the algorithm defines the sequence consisting only of the vector $z^0 = 1$, since $|N| - 1 = 0$. Hence, the output of the algorithm is the vector $v = (1,\ 2)^T$ which is harmonic. This example shows that Algorithm 1 can be arbitrarily faster than the more natural fixed point iteration method, which would compute the nonincreasing sequence $u^{(0)} = u$, $u^{(1)} = Au^{(0)}$, ..., until $u^{(k+1)} = u^{(k)}$. It is easy to see that the smallest k with this property is the smallest integer not less than $3/\epsilon$, and so, this alternative algorithm can be arbitrarily slow.

In order to apply Algorithm 1, we need to determine the critical graph of A. For all vectors $w \in \mathbb{R}^n$, define the *saturation graph* $\mathrm{Sat}(A, w)$ to be the set of arcs (i, j) such that $(Aw)_i = A_{ij} + w_j$. The critical graph can be determined thanks to the following result.

Proposition 3 ([1, Theorem. 3.98]). *Suppose that A is an $n \times n$ matrix with entries in $\mathbb{R} \cup \{-\infty\}$ such that every row of A has at least one finite entry, and $\chi(A) = 0$. Let $w \in \mathbb{R}^n$ be an arbitrary super-harmonic vector. Then, the circuits belonging to Sat(A, w) are precisely the critical circuits of A.*

The set of nodes belonging to the circuits of $\mathrm{Sat}(A, w)$ can be computed in a time which is linear in the number of arcs of $\mathrm{Sat}(A, w)$, by applying standard Tarjan-type depth first search algorithms. Hence,

Corollary 1. *Let A be as in Proposition 3. Then, the set of critical nodes of A can be computed in linear time from any super-harmonic vector of A.* $\square$

Remark 1. Assume that A is as in Proposition 3. Then, for any vector $b \in \mathbb{R}^n$, the vector A^*b is super-harmonic. Hence, a super-harmonic vector can be computed in $O(nm)$ time by the Ford-Bellman algorithm.

2.2 Spectral projectors acting on germs of affine functions

To handle invariant half-lines, it is convenient to use *germs of affine functions* at infinity, following [13]. Define the equivalence relation $\sim$ on the set of functions from $\mathbb{R}_+$ to $\mathbb{R}$, which is such that $u \sim v$ if $u(t) = v(t)$ for all t large enough. A *germ of affine function* is an equivalence class for the relation of $\sim$ containing an affine function. We denote by $\mathbb{G}$ the set of germs of affine functions. The usual relations and operations on functions trivially pass to germs. Hence, in the sequel, we denote by the same symbol a function and its germ, and we use the usual notation $\leq, +, \times$, etc., for the quotient relations or operations on germs. Denoting by ω the equivalence class of the identity map, we can write any germ of affine function as $a + b\omega$, where $a, b \in \mathbb{R}$, meaning the equivalence class of functions such that $t \mapsto a + bt$ for t large enough. We call

$$\ell(u) := \lim_{t\to\infty} u(t)/t = b$$

the *linear part* of u. The notation ω is justified by the fact that the maximum of the germs $a + b\omega$ and $a' + b'\omega$ corresponds to the lexicographic maximum of the vectors (b, a) and (b', a'), For instance, $\max(100 + 3\omega, 1 + 7\omega) = 1 + 7\omega$. So ω may be interpreted as a "sufficiently large number".

Let u denote a function from $\mathbb{R}_+$ to $\mathbb{R}^n$ which is affine for sufficiently large values of t. Since a min-max function f is piecewise affine, the function $f \circ u$ coincides with an affine function of t for sufficiently large t. It follows that f sends $\mathbb{G}^n$ to $\mathbb{G}^n$. The fact that u is an invariant half-line of f can be written as $f \circ u = \theta \circ u$, where θ is the shift operator, defined by $\theta \circ u(t) = u(t + 1)$.

If A is an arbitrary matrix with at least one finite entry per row, we define the *critical graph* of A to be the union of circuits γ of the digraph of $G(A)$ such that the weight-to-length ratio of γ is equal to $\chi_i(A)$, for any node i of γ. These circuits are called the *critical circuits*, and their nodes are called the *critical nodes*. When $\chi(A) = 0$, we recover the definitions of Section 2.1. We denote again by $C(A)$ the set of critical nodes of A.

We warn the reader that several nonequivalent definitions of the critical graph are relevant when the digraph of A is not strongly connected. In particular, in [8], the critical graph is defined as the union of the circuits with weight-to-length ratio $\rho_{\max}(A)$. This yields a smaller critical digraph, which is adapted to the max-plus eigenvector problem. The present critical graph is adapted to the invariant half-line problem, $f(u) = \theta \circ u$.

We extend the definition of the linear part ℓ to $\mathbb{G}^n$: if $u \in \mathbb{G}^n$, $\ell(u)$ denotes the vector with entries $\ell(u_i)$. In order to characterise the invariants half-lines of a max-plus linear map, we need the following elementary lemma.

Lemma 1. *Let $A \in (\mathbb{R} \cup \{-\infty\})^{n\times n}$ denote a matrix with at least one finite entry per row, and let $\eta := \chi(A)$. Then,*

$$\eta_i = \max_{A_{ij}\neq-\infty} \eta_j \enspace . \tag{6}$$

We denote by $S := S(A)$ the set of arcs (i, j) which attain the maximum in the right hand side of (6), for some $1 \leq i \leq n$. We denote by A^S the matrix such that

$$A^S_{ij} = \begin{cases} A_{ij} & \text{if } (i,j) \in S \\ -\infty & \text{otherwise,.} \end{cases}$$

and we define the matrix $\bar{A}$ such that

$$\bar{A}_{ij} = -\chi_i(A) + A^S_{ij} \enspace .$$

This definition is motivated by the following result.

Lemma 2. *Suppose that A is an $n \times n$ matrix with entries in $\mathbb{R} \cup \{-\infty\}$ such that every row of A has at least one finite entry, and let $\eta := \chi(A)$. For all vectors $v \in \mathbb{R}^n$, we have*

$$A(v + \eta\omega) = \bar{A}v + \eta(\omega + 1) \enspace .$$

Proposition 4. *Suppose that A is an $n \times n$ matrix with entries in $\mathbb{R} \cup \{-\infty\}$ such that every row of A has at least one finite entry, and let $v, \eta \in \mathbb{R}^n$, Then, $v + \eta\omega$ is an invariant half-line of A if and only if $\eta = \chi(A)$ and $\bar{A}v = v$.*

The following result generalises Proposition 2 to the case of max-plus operators with a non-zero cycle time.

Proposition 5 (Extension of the spectral projector to germs of affine functions). *Let $A \in (\mathbb{R} \cup \{-\infty\})^{n\times n}$ denote a matrix with at least one finite entry per row. Let $u \in \mathbb{G}^n$ be such that $A \circ u \leq \theta \circ u$ and $\ell(u) = \chi(A)$. Then, there is a unique $w \in \mathbb{G}^n$ such that $A \circ w = \theta \circ w$, and $u_i = w_i$ for all $i \in C := C(A)$. It satisfies $w \leq u$, and it can be determined as follows. Let us write u in the form $v + \eta\omega$, with $v, \eta \in \mathbb{R}^n$. Then, $w = z + \eta\omega$, where $z = P_{\bar{A}}v$, and $P_{\bar{A}}$ is the spectral projector of $\bar{A}$ defined in Definition 1.*

Definition 2 (Extension of the spectral projector). *We define the spectral projector of A extended to germs of affine functions, to be the map $u \mapsto w$, where u and w are as in the previous proposition. We still denote this map by P_A.*

Remark 2. Since $P_{\bar{A}}v$ can be computed in time $O(nm)$ and space $O(n)$, it follows from Proposition 5 that if $u \in \mathbb{G}^n$ is such that $A \circ u \leq \theta \circ u$ and $\ell(u) = \chi(A)$, then, $P_A u$ can be computed in time $O(nm)$ and space $O(n)$.

Example 2. Let

$$A = \begin{pmatrix} 0 & 2 & -\infty & -4 \\ 1 & -\infty & -1 & 0 \\ -7 & -\infty & 4 & -\infty \\ -\infty & -\infty & -\infty & 3 \end{pmatrix}$$

The graph of A has two strongly connected components, $U_1 = \{1,2,3\}$ and $U_2 = \{4\}$. We have $\rho_{\max}(A_{U_1U_1}) = 4$ and $\rho_{\max}(A_{U_2U_2}) = 3$. Since there is a path from every node of U_1 to the unique node of U_2, node 4, and since node 4 has access only to itself, we deduce from Proposition 1 that

$$\chi(A) = (4,\ 4,\ 4,\ 3)^T \ .$$

The critical graph of A consists of the circuits $3 \to 3$ and $4 \to 4$. Hence, the set of critical nodes is $C = \{3,4\}$. Let $v = (2,\ 1\ -5,\ -8)^T$, and consider the germ

$$u = v + \chi(A)\omega = (2+4\omega,\ 1+4\omega,\ -5+4\omega,\ -8+3\omega)^T \ .$$

We have

$$\begin{aligned} Au &= (3+4\omega,\ 3+4\omega,\ -1+4\omega,\ -5+3\omega)^T \\ &\leq \theta\circ u = (6+4\omega,\ 5+4\omega,\ -1+4\omega,\ -5+3\omega)^T \ . \end{aligned}$$

For instance, the first entry of Au is obtained from $(Au)_1 = (0+2+4\omega) \vee (2+1+4\omega) \vee (-4-8+3\omega) = 3+4\omega$.

The graph S consists of the arcs (i,j) in $G(A)$, with $i,j \in U_1$ or $i = j = 4 \in U_2$. So,

$$A^S = \begin{pmatrix} 0 & 2 & -\infty & -\infty \\ 1 & -\infty & -1 & -\infty \\ -7 & -\infty & 4 & -\infty \\ -\infty & -\infty & -\infty & 3 \end{pmatrix}$$

and

$$\bar{A} = \begin{pmatrix} -4 & -2 & -\infty & -\infty \\ -3 & -\infty & -5 & -\infty \\ -11 & -\infty & 0 & -\infty \\ -\infty & -\infty & -\infty & 0 \end{pmatrix}$$

We have $z_N = \bar{A}^*_{NN}\bar{A}_{NC}v_C$, where $v_C = (-5,\ -8)^T$ and

$$\bar{A}_{NN} = \begin{pmatrix} -4 & -2 \\ -3 & -\infty \end{pmatrix}, \qquad \bar{A}_{NC} = \begin{pmatrix} -\infty & -\infty \\ -5 & -\infty \end{pmatrix}$$

The vector z_N can be computed by applying steps 2 and 3 of Algorithm 1. We have

$$\begin{aligned} z^{(0)} &= b = \bar{A}_{NC}v_C = (-\infty,\ -10)^T, \\ z^{(1)} &= \bar{A}_{NN}z^{(0)} \vee b = (-12,\ -10)^T \end{aligned}$$

and since $|N| = 2$, the algorithm stops, returning

$$w := P_A u = (-12+4\omega,\ -10+4\omega,\ -5+4\omega,\ -8+3\omega)^T \ .$$

We know from Proposition 5, and it can be checked directly, that

$$Aw = (-8+4\omega,\ -6+4\omega,\ -1+4\omega,\ -5+3\omega)^T = \theta\circ w \ .$$

3. POLICY ITERATION ALGORITHM FOR ZERO-SUM TWO PLAYERS DETERMINISTIC GAMES WITH MEAN PAYOFF

3.1 The algorithm

We now present a variant of the policy iteration algorithm of [13], in which a new evaluation of the spectral projector for germs, using Proposition 5 and Algorithm 1, as well as a simplification, are incorporated.

Recall that to any (feedback, stationary) strategy σ of Min is associated the map $f^{(\sigma)}$ defined by (4). Similarly, the map $g^{(\sigma)}$ from $\mathbb{R}^p$ to $\mathbb{R}^n$ is defined by

$$g_j^{(\sigma)}(x) = r_{j'\sigma(j')} + x_{\sigma(j')} \ ,$$

for $j = 1, \ldots, n$, so that

$$f^{(\sigma)} = g^{(\sigma)}\circ g' \ .$$

Algorithm 2 (Min-max policy iteration algorithm, see [13]). Input: A weighted bipartite digraph, such that every node is the tail of at least one arc. We denote by f the associated dynamic programming operator, which is defined by as in Section 1.2. *Output*: An invariant half-line of f, that is, $u \in \mathbb{G}^n$ such that $f\circ u = \theta\circ u$.

1. *Initialisation.* Select an arbitrary policy of Min, σ_1. Compute an invariant half-line of $f^{(\sigma_1)}$, that is, $u^{(1)} \in \mathbb{G}^n$ such that $f^{(\sigma_1)}\circ u^{(1)} = \theta\circ u^{(1)}$. Set $k = 1$.

2. Evaluate $f\circ u^{(k)} \in \mathbb{G}^n$. If $f\circ u^{(k)} = f^{(\sigma_k)}\circ u^{(k)}$, $u^{(k)}$ is an invariant half-line of f, stop.

3. Otherwise, we *improve the policy*, by selecting a strategy σ_{k+1} of Min such that
$$f\circ u^{(k)} = f^{(\sigma_{k+1})}\circ u^{(k)} \ .$$
The strategy σ_{k+1} is chosen so that it keeps the values of σ_k, whenever possible, meaning that $\sigma_{k+1}(j') = \sigma_k(j')$ if $f_j\circ u^{(k)} = f_j^{(\sigma_k)}\circ u^{(k)}$.

4. *Value determination.* We compute an arbitrary invariant half-line $v \in \mathbb{G}^n$ of $f^{(\sigma_{k+1})}$. If $\ell(v) \neq \ell(u^{(k)})$, we set $u^{(k+1)} := v$. If $\ell(v) = \ell(u^{(k)})$, we say that the iteration is *degenerate*, and we choose for $u^{(k+1)}$ the image of the germ $u^{(k)}$ by the spectral projector of the map $f^{(\sigma_{k+1})}$ (Definition 2).

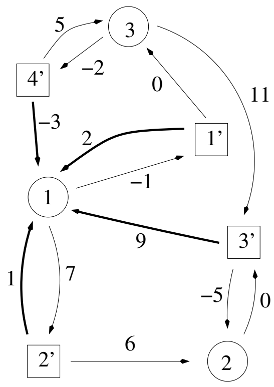
5. We increment k by one and go to step 2.

The correctness of the algorithm can be proved by combining the arguments of the proof of Theorem 2.2 in [8] with Proposition 5 above. We shall not give the proof here.

By comparison with the algorithmm of [13], Algorithm 2 incorporates the fast evaluation of the spectral projector (Proposition 5 and Remark 2). Additionnaly, the policy improvement rule in step 3 has been slightly simplified, to make the algorithm more transparent (unlike the computation of the spectral projector, this modification does not seem to have any significant effect on the speed of the method).

3.2 Illustration

Let us apply Algorithm 2 to the game of Fig. 1. We start from the strategy $\sigma_1(i') = 1$, $i = 1, \ldots, 4$, which determines the moves represented in bold on the following graph.



The max-plus linear operator associated to σ_1 is given by

$$f^{(\sigma_1)}(x) = \begin{pmatrix} (1+x_1)\vee(9+x_2) \\ x_1 \vee (8+x_2) \\ (8+x_1)\vee(16+x_2) \\ (-4+x_1)\vee(4+x_2) \end{pmatrix} .$$

An invariant half-line of $f^{(\sigma_1)}$ can be easily computed (see [6] for details). We can take:

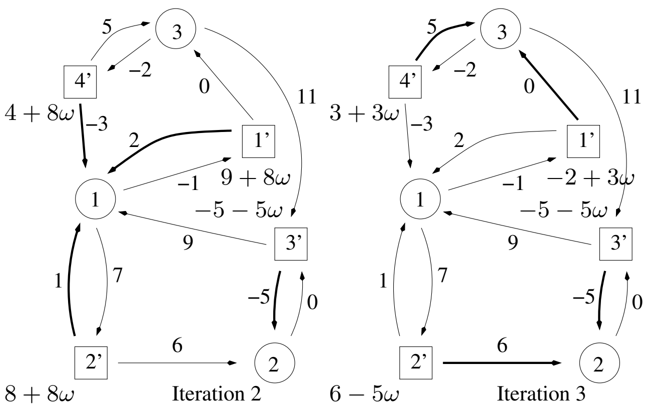
$$u^{(1)} = (9+8\omega,\ 8+8\omega,\ 16+8\omega,\ 4+8\omega)^T .$$

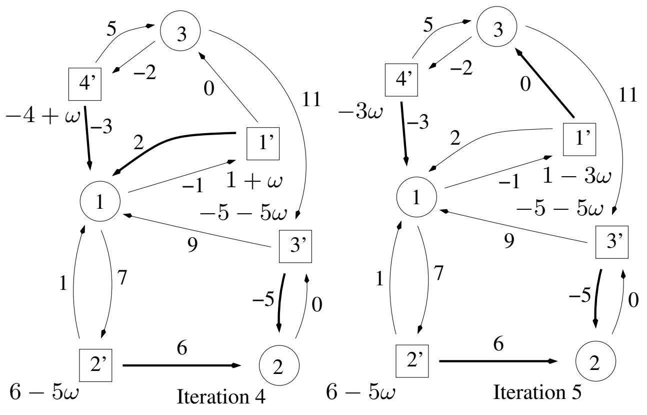
Now, in step 2, we evaluate $g' \circ u^{(1)}$, and then $f \circ u^1 = g \circ g' \circ u^1$:

$$g' \circ u^{(1)} = (15+8\omega,\ 16+8\omega,\ 27+8\omega)^T ,$$

$$f \circ u^{(1)} = \begin{pmatrix} \underline{(2+15+8\omega)} \wedge (27+8\omega) \\ \underline{(1+15+8\omega)} \wedge (6+16+8\omega) \\ \underline{(9+15+8\omega)} \wedge \underline{(-5+16+8\omega)} \\ \underline{(-3+8+8\omega)} \wedge (5+27+8\omega) \end{pmatrix} = \begin{pmatrix} 17+8\omega \\ 16+8\omega \\ 11+8\omega \\ 12+8\omega \end{pmatrix} .$$

The terms which realise the minimum are underlined. We have $f_3 \circ u^{(1)} < f_3^{\sigma_1} \circ u^{(1)}$, and so, we improve the strategy (Step 3). We take for σ_2 the unique strategy which realises the minimum above, $\sigma_2(1') = \sigma_2(2') = \sigma_2(4') = 1$, $\sigma_2(3') = 2$. The next iterations are carried out in the same way. In this example, all the iterations turn out to be nondegenerate. We only show the graphs corresponding to the successive policies, together with the vectors $u^{(k)}$:





Every entry $u_i^{(k)}$ is indicated near node i' at iteration k. The algorithm terminates at Iteration 5, with the invariant half-line of f which can be read on the graph:

$$v = (1-3\omega,\ 6-5\omega,\ -5-5\omega,\ -3\omega)^T .$$

This shows that $\chi(f) = (-3,\ -5,\ -5,\ -3)^T$, confirming the initial analysis of Section 1.1.

4. EXPERIMENTAL RESULTS

We now present experimental results. We considered random graphs, then a simple pursuit-evasion game, and finally, a concrete problem arising in the control of discrete event systems, which is taken from a work of Katz [20].

Let us now describe the details of implementation of Algorithm 2. Invariant half-lines of max-plus linear maps were computed using the policy iteration algorithm of [6] (which is experimentally more efficient than the method based on Karp's algorithm [19]). We used floating point arithmetics, so the equality tests in Algorithm 2 and in Algorithm 4.4 of [6] have to be understood up to a fixed ϵ parameter, which was chosen to be $\epsilon := \epsilon' \times (\max(M', M) + 1)$ where

$$M' = \max_{(i,j')\in E} r_{ij'} - \min_{(i,j')\in E} r_{ij'},\ M = \max_{(j',k)\in E} r_{j'k} - \min_{(j'k)\in E} r_{j'k}$$

and ϵ' is a small constant (the real numbers were coded with a double precision, and we took $\epsilon' = 10^{-12}$).

The algorithm has been programmed in C++ , and compiled using the C++ compliant gcc version 3.3.3 without using any optimisation option. The experiments have been conducted on an Intel Pentium 4 processor at 3 GHz, with 1 GB of RAM. The operating system used was linux kernel 2.6.5-1.358.

4.1 Experiments on random graphs

4.1.1 Complete bipartite graphs

We tested the algorithm on complete bipartite graphs, in which $n = p$. Such graphs have $2n$ nodes and $m = 2n^2$ arcs. The weights of the arcs were integers drawn at random with the uniform distribution in the range $[0, 1000]$ (we also made tests with real weights, taken with the uniform distribution in an interval, and obtained similar results). In order to guarantee the reliability of the experiments, we used the uniform random number generator “Mersenne twister” [22].

We denote by $N_{\min}$ the number of iterations of Algorithm 2, that is, the number of strategies chosen by Min before the algorithm terminates. For instance, for the game of section 3.2, $N_{\min} = 5$. For each dimension n from 1 to

1600, we tested a sample of 100 graphs. Fig. 2 represents the number of iterations of Min, as a function of the dimension n. To appreciate the randomness of $N_{\min}$, we displayed the average number, maximal number, and minimal value of $N_{\min}$ over the sample. The figure indicates that the number of iterations grows very slowly with the dimension.

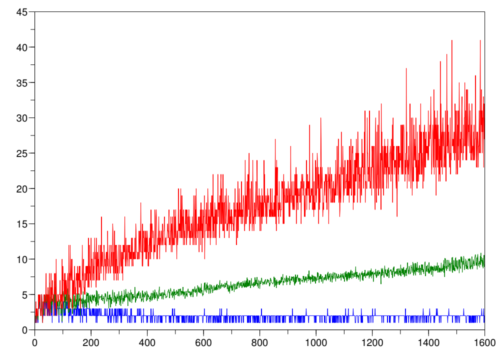


Figure 2: Number of iterations of minimizer $N_{\min}$ as a function of n for complete bipartite graphs. The red (top), green (middle), and blue (bottom) curves represent respectively the maximum, average, and minimum value of $N_{\min}$, computed on a sample of 100 **graphs, for each dimension.**

In order to estimate the execution time in a machine independent way, we counted the *global number of iterations*, N_{glob}, which is the cumulated number of iterations of the algorithm of [6] which is called by Algorithm 2 to compute invariant half-lines, plus the cumulated number of iterations of the Ford-Bellman algorithm which is called when computing the spectral projector. Every global iteration takes a time asymptotically proportional to m, so the total execution time is asymptotically proportional to $N_{\text{glob}} \times m$. The global number of iterations N_{glob} is displayed in Fig. 3, for the same samples as in Fig. 2. The execution time is

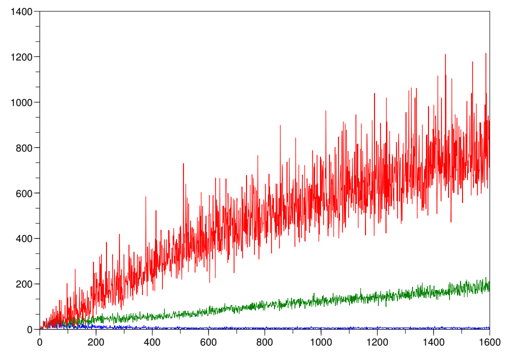


Figure 3: Number of global iterations N_{glob}, as a function of n. Same conventions as in Fig. 2.

shown in Fig. 4. We denote by N_{spec} the number of degenerate iterations, in which the routine computing the spectral projector is called. Table 1 gives some values of the minimal, average, and maximal value of $N_{\min}$, N_{glob}, and N_{spec}, computed from the same set of experimental data as the one used to produce the figures (in particular, the Min, Average,

Table 1: Experiments on complete bipartite graphs

Nodes	$N_{\min}$			N_{glob}			N_{spec}		
	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
500	2	4.95	13	7	57.7	331	1	3.06	8
1000	2	7.55	18	7	128.91	479	1	5.02	17
1500	2	8.69	28	4	164.66	794	0	6.69	27
2000	2	12.06	35	7	238.28	1021	1	9.85	34
2500	2	18.64	52	7	378.22	1202	1	16.42	51
3000	2	22.07	62	7	318.78	1410	1	20.36	60

From	To	Step Size
2	5000	1
5100	50000	100
50500	200000	500
201000	400000	1000

Table 2: Step size for the experiments of Fig. 5–7.

and Max values in Table 1 refer to a sample of 100 graphs for each value of n). A significant proportion of iterations are degenerate.

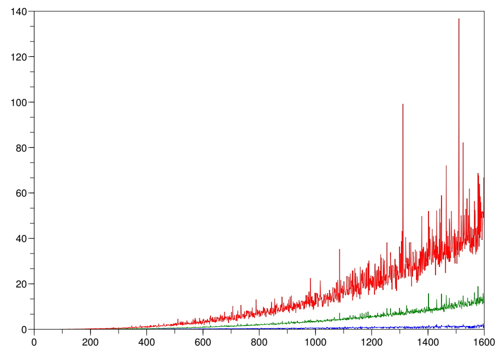


Figure 4: CPU time (in seconds) taken by the algorithm as a function of n. Same conventions as in Fig. 2.

4.1.2 *Sparse bipartite graphs*

We considered sparse random bipartite graphs, with n nodes of each kind, such that every node has exactly 2 successors. So the number of nodes is $2n$, and the number of arcs is $4n$. The two elements set of successors of a given node was drawn with the uniform distribution. We made the number n vary as shown in Table 2, testing a sample of 10 graphs for each value of n. The numbers of iterations $N_{\min}$, N_{glob}, and the CPU time, are shown on Fig. 5–7. Typical values of $N_{\min}$, N_{glob}, and N_{spec} are given in Table 3. This indicates that the growth of the number of iterations is *sublinear* in n.

4.2 Cat and mouse in a room with an obstacle

In this section, we present experiments concerning a pursuit evasion game. Suppose there is a cat and a mouse in a room. The following assumptions are made: 1) The mouse (Min) wishes to minimise the negative of its distance to the cat, and the cat (Max) tries to maximise the negative of its distance to the mouse. This is modelled by assuming that the payments occur only after the cat's actions. Then, the

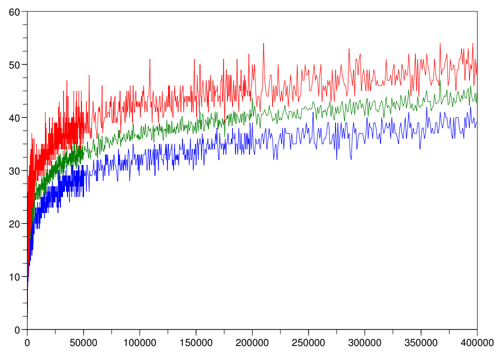


Figure 5: Number of iterations of minimizer $N_{\min}$ as a function of n, for random sparse bipartite graphs with outdegree 2. The red (top), green (middle), and blue (bottom) curves represent respectively the maximum, average, and minimum value of $N_{\min}$, computed on a sample of 10 graphs, for each dimension.

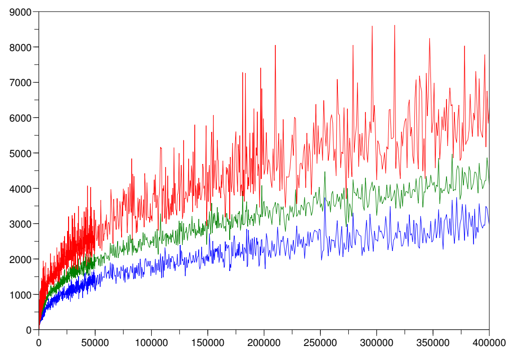


Figure 6: Number of global iterations N_{glob} as a function of n, for random sparse bipartite graphs with outdegree 2. Same conventions as in Fig. 5.

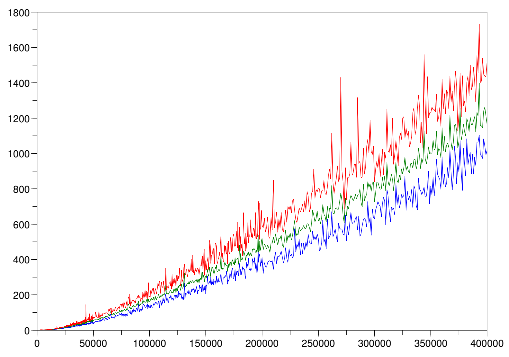


Figure 7: CPU time (in seconds) as a function of n, for random sparse bipartite graphs with outdegree 2. Same conventions as in Fig. 5.

Table 3: Sparse random bipartite graphs with outdegree 2

Nodes	$N_{\min}$			N_{glob}			N_{spec}		
	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
500000	38	44.6	54	2632	4207.1	5611	2	3.9	6
1000000	46	48.5	52	4036	6378.1	9560	2	5.3	9
2000000	47	49.5	55	5255	7538.5	10174	3	4.6	7
3000000	50	53.2	57	5329	8419.5	13276	2	4.6	7
4000000	47	55.3	63	5396	8993.4	13283	1	4.2	7
5000000	51	55.2	63	7699	11237.6	14094	3	5.3	8
6000000	52	57.6	65	7155	12370.3	19617	2	5.2	10

mouse receives the negative of its distance to the cat. 2) The mouse makes the first action, and then, the cat and mouse alternate their actions. 3) An action consists of moving to a neighbour place, or staying at the same place.

We considered a square room, of size $\ell \times \ell$, with ℓ even. To make the game more interesting, we put an obstacle, i.e., a zone to which both animals cannot gain access to. (Without obstacle, the cat ultimately reaches the mouse.) This is illustrated by the following example, in which $\ell = 4$:

1	2	3	4
5	**ob**	**st**	6
7	**ac**	**le**	8
9	10	11	12

The following table gives the final distance (FD) between the cat and mouse, as a function of their initial position (C and M), assuming both animals move optimally. The set of initial positions given in the table is exhaustive, up to symmetry. We chose the distance induced by the sup-norm.

C	M	FD	C	M	FD	C	M	FD	C	M	FD
1	1	0	2	1	0	3	1	1	4	1	2
1	2	1	2	2	0	3	2	1	4	2	2
1	3	2	2	3	1	3	3	0	4	3	1
1	4	2	2	4	1	3	4	0	4	4	0
1	5	1	2	5	1	3	5	2	4	5	3
1	6	3	2	6	2	3	6	1	4	6	1
1	7	2	2	7	2	3	7	3	4	7	3
1	8	3	2	8	3	3	8	2	4	8	2
1	9	2	2	9	2	3	9	3	4	9	3
1	10	3	2	10	3	3	10	3	4	10	3
1	11	3	2	11	3	3	11	3	4	11	3
1	12	3	2	12	3	3	12	2	4	12	2

We generated the bipartite graph modelling this game. The number of nodes and the number of arcs are both of order ℓ^4. Table 4 lists some of the experiments conducted for this case. The CPU time does not include the time taken to generate the bipartite graph. The number of iterations $N_{\min}$ is of the same order as the diameter of the room (i.e., it is of order $n^{1/4}$, where n is the number of nodes of the bipartite graph representing the game).

4.3 The $Ax = Bx$ problem

One motivation for the cycle time problem for min-max functions is to solve systems of max-plus linear equations. Indeed, if A and B are $n \times p$ matrices with entries in $\mathbb{R} \cup \{-\infty\}$, the max-plus linear problem $Ax = Bx$ can be rewritten equivalently as $Ax \leq Bx$ and $Bx \leq Ax$ or

$$x_i \leq \min\Big(\min_{j:\, A_{ji} > -\infty} (-A_{ji} + \max_k B_{jk} + x_k),$$
$$\min_{j:\, B_{ji} > -\infty} (-B_{ji} + \max_k A_{jk} + x_k)\Big) \ .$$

Table 4: Cat and Mouse Problem

ℓ	$N_{\min}$	N_{glob}	N_{spec}	CPU time (sec.)
4	5	26	2	0
6	11	141	8	0.09
8	20	396	15	0.97
10	24	654	19	4.55
12	28	938	22	13.75
14	28	1086	21	29.86
16	35	1615	28	80.88
18	39	2069	31	167.57
20	39	2363	31	289.66
24	48	3363	36	1015.11
30	55	4733	39	3123.41
36	69	7951	54	11274.94

This is a sub-fixed point problem of the form $x \leq f(x)$. Under mild conditions (for instance, if both A and B have at least one finite entry in each column and in each row), the map f sends $\mathbb{R}^n$ to $\mathbb{R}^n$, and so, it is a min-max function. It is easy to see that $Ax = Bx$ has a finite solution x if and only if $\chi(f) \geq 0$. Indeed, if $\chi(f) \geq 0$, we can take $x = v + t\eta$, where $t \mapsto v + t\eta$ is an invariant half-line of f, and t is large enough. The algorithm was tested on the feedback synthesis problem studied by Katz in [20], which leads to a system of the form $Ax = Bx$ with 58 unknowns and 40 equations. The policy iteration algorithm took 3 iterations, in which the spectral projector was called twice.

Acknowledgments

The first author was supported by the French embassy in New Delhi through the INRIA international internship programme. He thanks Ricardo Katz for his help. The second author thanks Thierry Bousch for having pointed out to him the work of Zwick and Paterson [25] and the works on parity games (see Vöge and Jurdziński [24]).

5. REFERENCES

[1] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat. *Synchronization and Linearity.* Wiley, 1992.

[2] T. Bewley and E. Kohlberg. The asymptotic solution of a recursion equation occurring in stochastic games. *Math. Oper. Res.*, 1(4):321–336, 1976.

[3] H. Bjorklund, S. Sandberg, and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. Technical Report 05, DIMACS, 2004.

[4] S. Burns. *Performance Analysis and Optimization of Asynchronous Circuits.* Phd thesis, California Institute of Technology, 1990.

[5] Y. Cheng and D.-Z. Zheng. A cycle time computing algorithm and its application in the structural analysis of min-max systems. *Discrete Event Dyn. Syst.*, 14(1):5–30, 2004.

[6] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. M. Gettrick, and J.-P. Quadrat. Numerical computation of spectral elements in max-plus algebra. In *Proc. of the IFAC Conference on System Structure and Control*, Nantes, July 1998.

[7] J. Cochet-Terrasson and S. Gaubert. A min-max analogue of the Howard algorithm. Privately circuled draft, 1996.

[8] J. Cochet-Terrasson, S. Gaubert, and J. Gunawardena. A constructive fixed point theorem for min-max functions. *Dynamics and Stability of Systems*, 14(4):407–433, 1999.

[9] G. Cohen, D. Dubois, J. Quadrat, and M. Viot. Analyse du comportement périodique des systèmes de production par la théorie des dioïdes. Rapport de recherche 191, INRIA, Le Chesnay, France, 1983.

[10] A. Condon. The complexity of stochastic games. *Inform. and Comput.*, 96(2):203–224, 1992.

[11] A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A policy iteration algorithm for computing fixed points in static analysis of programs. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*, number 3576 in LNCS, pages 462–475, Edinburgh, July 2005. Springer.

[12] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *Internat. J. Game Theory*, 8(2):109–113, 1979.

[13] S. Gaubert and J. Gunawardena. The duality theorem for min-max functions. *C.R. Acad. Sci.*, 326(1):43–48, 1998.

[14] J. Gunawardena. Min-max functions. *Discrete Event Dynamic Systems*, 4:377–406, 1994.

[15] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *U.S.S.R. Comput. Math. Phys.*, 28(5):85–91, 1988.

[16] A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management sciences*, 12(5):359–370, 1966.

[17] R. Howard. *Dynamic Programming and Markov Processes.* Wiley, 1960.

[18] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, January 2006.

[19] R. Karp. A characterization of the minimum mean-cycle in a digraph. *Discrete Maths.*, 23:309–311, 1978.

[20] R. D. Katz. Max-plus (A,B)-invariant spaces and control of timed discrete event systems. 2005. e-print arXiv:math.OC/0503448, to appear in IEEE-TAC.

[21] E. Kohlberg. Invariant half-lines of nonexpansive piecewise-linear transformations. *Math. Oper. Res.*, 5(3):366–372, 1980.

[22] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[23] G. Olsder. Eigenvalues of dynamic min-max systems. *J. of Discrete Event Dynamic Systems*, 1:177–207, 1991.

[24] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proceedings of 12th Int. Conf. on Computer Aided Verification (CAV'2000)*, July 2000.

[25] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoret. Comput. Sci.*, 158(1-2):343–359, 1996.