# To Reach or not to Reach? Efficient Algorithms for Total-Payoff Games*

**Thomas Brihaye[1], Gilles Geeraerts[2], Axel Haddad[1], and Benjamin Monmege[2]**

1 **Université de Mons, Belgium,** thomas.brihaye,axel.haddad@umons.ac.be
2 **Université libre de Bruxelles, Belgium,** gigeerae,benjamin.monmege@ulb.ac.be

**Abstract**

Quantitative games are two-player zero-sum games played on directed weighted graphs. Total-payoff games – that can be seen as a refinement of the well-studied mean-payoff games – are the variant where the payoff of a play is computed as the sum of the weights. Our aim is to describe the first pseudo-polynomial time algorithm for total-payoff games in the presence of arbitrary weights. It consists of a non-trivial application of the value iteration paradigm. Indeed, it requires to study, as a milestone, a refinement of these games, called min-cost reachability games, where we add a reachability objective to one of the players. For these games, we give an efficient value iteration algorithm to compute the values and optimal strategies (when they exist), that runs in pseudo-polynomial time. We also propose heuristics to speed up the computations.

**1998 ACM Subject Classification** D.2.4 Software/Program Verification, F.3.1 Specifying and Verifying and Reasoning about Programs

**Keywords and phrases** Games on graphs, Reachability, Quantitative games, Value iteration

**Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2015.297

## 1 Introduction

*Games played on graphs* are nowadays a well-studied and well-established model for the computer-aided design of computer systems, as they enable *automatic synthesis* of systems that are *correct-by-construction*. Of particular interest are *quantitative games*, that allow one to model precisely *quantitative* parameters of the system, such as energy consumption. In this setting, the game is played by two players on a directed weighted graph, where the edge weights model, for instance, a cost or a reward associated to the moves of the players. Each vertex of the graph belongs to one of the two players who compete by moving a token along the graph edges, thereby forming an infinite path called a *play*. With each play is associated a real-valued *payoff* computed from the sequence of edge weights along the play. The traditional payoffs that have been considered in the literature include total-payoff [10], mean-payoff [7] and discounted-payoff [17]. In this quantitative setting, one player aims at maximising the payoff while the other tries to minimise it. So one wants to compute, for each player, the best payoff that he can guarantee from each vertex, and the associated optimal strategies (i.e., that guarantee the optimal payoff no matter how the adversary is playing).

Such quantitative games have been extensively studied in the literature. Their associated decision problems (*is the value of a given vertex above a given threshold?*) are known to be

* The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under Grant Agreement no601148 (CASSTING).



in NP ∩ co-NP. Mean-payoff games have arguably been best studied from the algorithmic point of view. A landmark is Zwick and Paterson's pseudo-polynomial time (i.e., polynomial in the weighted graph when weights are encoded in unary) algorithm [17], using the *value iteration* paradigm that consists in computing a sequence of vectors of values that converges towards the optimal values of the vertices. After a fixed, pseudo-polynomial, number of steps, the computed values are precise enough to deduce the actual values of all vertices. Better pseudo-polynomial time algorithms have later been proposed, e.g., in [1, 4, 6], also achieving sub-exponential expected running time by means of randomisation.

In this paper, we focus on *total-payoff games*. Given an infinite play $\pi$, we denote by $\pi[k]$ the prefix of $\pi$ of length $k$, and by $\mathbf{TP}(\pi[k])$ the (finite) sum of all edge weights along this prefix. The *total-payoff* of $\pi$, $\mathbf{TP}(\pi)$, is the inferior limit of all those sums, i.e., $\mathbf{TP}(\pi) = \liminf_{k\to\infty} \mathbf{TP}(\pi[k])$. Compared to mean-payoff (and discounted-payoff) games, the literature on total-payoff games is less extensive. Gimbert and Zielonka have shown [10] that optimal memoryless strategies always exist for both players and the best algorithm to compute the values runs in exponential time [9], and consists in iteratively improving strategies. Other related works include *energy games* where one player tries to optimise its energy consumption (computed again as a sum), keeping the energy level always above 0 (which makes difficult to apply techniques solving those games in the case of total-payoff); and a probabilistic variant of total-payoff games, where the weights are restricted to be non-negative [5]. Yet, we argue that the total-payoff objective is interesting as a *refinement* of the mean-payoff. Indeed, recall first that the total-payoff is finite if and only if the mean-payoff is null. Then, the computation of the total-payoff enables a finer, two-stage analysis of a game $\mathcal{G}$: (i) compute the mean payoff $\mathbf{MP}(\mathcal{G})$; (ii) subtract $\mathbf{MP}(\mathcal{G})$ from all edge weights, and scale the resulting weights if necessary to obtain integers. At that point, one has obtained a new game $\mathcal{G}'$ with null mean-payoff; (iii) compute $\mathbf{TP}(\mathcal{G}')$ to *quantify the amount of fluctuation around the mean-payoff* of the original game. Unfortunately, so far, no efficient (i.e., pseudo-polynomial time) algorithms for total-payoff games have been proposed, and straightforward adaptations of Zwick and Paterson's value iteration algorithm for mean-payoff do not work, as we demonstrate at the end of Section 2. In the present article, we fill in this gap by introducing the first pseudo-polynomial time algorithm for computing the values in total-payoff games.

Our solution is a non-trivial value iteration algorithm that proceeds through nested fixed points (see Algorithm 2). A play of a total-payoff game is infinite by essence. We transform the game so that one of the players (the minimiser) must ensure a *reachability objective*: we assume that the game ends once this reachability objective has been met. The intuition behind this transformation, that stems from the use of an inferior limit in the definition of the total-payoff, is as follows: in any play $\pi$ whose total-payoff is *finite*, there is a position $\ell$ in the play after which all the partial sums $\mathbf{TP}(\pi[i])$ (with $i \geqslant \ell$) will be larger than or equal to the total-payoff $\mathbf{TP}(\pi)$ of $\pi$, and infinitely often both will be equal. For example, consider the game depicted in Figure 1(a), where the maximiser player (henceforth called Max) plays with the round vertices and the minimiser (Min) with the square vertices. For both players, the optimal value when playing from $v_1$ is 2, and the play $\pi = v_1v_2v_3\ v_4v_5\ v_4v_3\ (v_4v_5)^\omega$ reaches this value (i.e., $\mathbf{TP}(\pi) = 2$). Moreover, for all $k \geqslant 7$: $\mathbf{TP}(\pi[k]) \geqslant \mathbf{TP}(\pi)$, and infinitely many prefixes ($\pi[8]$, $\pi[10]$, $\pi[12]$, ...) have a total-payoff of 2, as shown in Figure 1(b).

Based on this observation, we transform a total-payoff game $\mathcal{G}$, into a new game that has *the same value as the original total-payoff game* but incorporates a reachability objective for Min. Intuitively, in this new game, we allow a new action for Min: after each play prefix $\pi[k]$, he can ask to *stop the game*, in which case the payoff of the play is the payoff $\mathbf{TP}(\pi[k])$

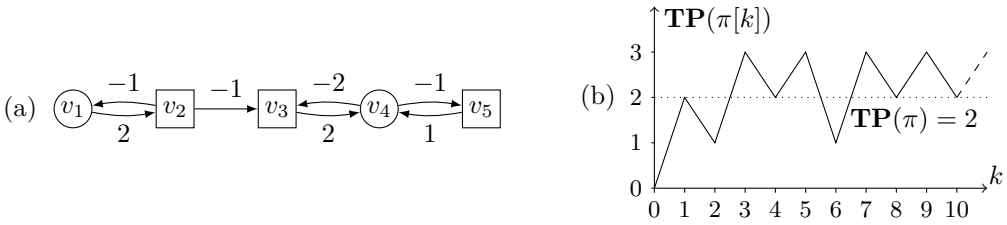


**Figure 1** (a) A total-payoff game, and (b) the evolution of the partial sums in $\pi$.

of the prefix. However, allowing Min to stop the game at any moment would not allow to obtain the same value as in the original total-payoff game: for instance, in the example of Figure 1(a), Min could secure value 1 by asking to stop after $\pi[2]$, which is strictly smaller that the actual total-payoff (2) of the whole play $\pi$. So, we allow Max to *veto* to stop the game, in which case both must go on playing. Again, allowing Max to turn down all of Min's requests would be unfair, so we parametrise the game with a natural number $K$, which is the maximal number of vetoes that Max can play (and we denote by $\mathcal{G}^K$ the resulting game). For the *play* depicted in Figure 1(b), letting $K = 3$ is sufficient: trying to obtain a better payoff than the optimal, Min could request to stop after $\pi[0]$, $\pi[2]$ and $\pi[6]$, and Max can veto these three requests. After that, Max can safely accept the next request of Min, since the total payoff of all prefixes $\pi[k]$ with $k \geqslant 6$ are larger than or equal to $\mathbf{TP}(\pi) = 2$. Our key technical contribution is to show that *for all total-payoff games, there exists a finite, pseudo-polynomial, value of $K$ such that the values in $\mathcal{G}^K$ and $\mathcal{G}$ coincide* (assuming all values are finite in $\mathcal{G}$: we treat the $+\infty$ and $-\infty$ values separately). Now, assume that, when Max accepts to stop the game (possibly because he has exhausted the maximal number $K$ of vetoes), the game moves to a *target state*, and stops. By doing so, we effectively reduce the computation of the values in the total-payoff game $\mathcal{G}$ to the computation of the values in the total-payoff game $\mathcal{G}^K$ *with an additional reachability objective* (the target state) for Min.

In the following, such refined total-payoff games – where Min *must* reach a designated target vertex – will be called *min-cost reachability games*. Failing to reach the target vertices is the worst situation for Min, so the payoff of all plays that do not reach the target is $+\infty$, irrespective of the weights along the play. Otherwise, the payoff of a play is the sum of the weights up to the first occurrence of the target. As such, this problem nicely generalises the classical shortest path problem in a weighted graph. In the one-player setting (considering the point of view of Min for instance), this problem can be solved in polynomial time by Dijkstra's and Floyd-Warshall's algorithms when the weights are non-negative and arbitrary, respectively. In [11], Khachiyan *et al.* propose an extension of Dijkstra's algorithm to handle the two-player, non-negative weights case. However, in our more general setting (two players, arbitrary weights), this problem has, as far as we know, not been studied as such, except that the associated decision problem is known to be in NP ∩ co-NP [8]. A pseudo-polynomial time algorithm to solve a very close problem, called the *longest shortest path problem* has been introduced by Björklund and Vorobyov [1] to eventually solve mean-payoff games. However, because of this peculiar context of mean-payoff games, their definition of the length of a path differs from our definition of the payoff and their algorithm cannot be easily adapted to solve our min-cost reachability problem. Thus, as a second contribution, we show that a value iteration algorithm enables us to compute in pseudo-polynomial time the values of a min-cost reachability game. We believe that min-cost reachability games bear their own

potential theoretical and practical applications[1]. Those games are discussed in Section 3. In addition to the pseudo-polynomial time algorithm to compute the values, we show how to compute optimal strategies for both players and characterise them: there is always a memoryless strategy for the maximiser player, but we exhibit an example (see Figure 2(*a*)) where the minimiser player needs (finite) memory. Those results on min-cost reachability games are exploited in Section 4 where we introduce and prove correct our efficient algorithm for total-payoff games.

Finally, we briefly present our implementation in Section 5, using as a core the numerical model-checker PRISM. This allows us to describe some heuristics able to improve the practical performances of our algorithms for total-payoff games and min-cost reachability games on certain subclasses of graphs. More technical explanations and full proofs may be found in an extended version of this article [2].

## 2 Quantitative games with arbitrary weights

We denote by $\mathbb{Z}$ the set of integers, and $\mathbb{Z}_\infty = \mathbb{Z} \cup \{-\infty, +\infty\}$. The set of vectors indexed by $V$ with values in $S$ is denoted by $S^V$. We let $\preccurlyeq$ be the pointwise order over $\mathbb{Z}_\infty^V$, where $x \preccurlyeq y$ if and only if $x(v) \leqslant y(v)$ for all $v \in V$.

We consider two-player turn-based games on weighted graphs and denote the two *players* by $\mathsf{Max}$ and $\mathsf{Min}$. A *weighted graph* is a tuple $\langle V, E, \omega \rangle$ where $V = V_{\mathsf{Max}} \uplus V_{\mathsf{Min}}$ is a finite set of vertices partitioned into the sets $V_{\mathsf{Max}}$ and $V_{\mathsf{Min}}$ of $\mathsf{Max}$ and $\mathsf{Min}$ respectively, $E \subseteq V \times V$ is a set of *directed edges*, $\omega \colon E \to \mathbb{Z}$ is the *weight function*, associating an integer weight with each edge. In our drawings, $\mathsf{Max}$ vertices are depicted by circles; $\mathsf{Min}$ vertices by boxes. For every vertex $v \in V$, the set of successors of $v$ by $E$ is denoted by $E(v) = \{v' \in V \mid (v, v') \in E\}$. Without loss of generality, we assume that every graph is deadlock-free, i.e., for all vertices $v$, $E(v) \neq \emptyset$. Finally, throughout this article, we let $W = \max_{(v,v') \in E} |\omega(v, v')|$ be the greatest edge weight (in absolute value) in the game graph. A *finite play* is a finite sequence of vertices $\pi = v_0 v_1 \cdots v_k$ such that for all $0 \leqslant i < k$, $(v_i, v_{i+1}) \in E$. A *play* is an infinite sequence of vertices $\pi = v_0 v_1 \cdots$ such that every finite prefix $v_0 \cdots v_k$, denoted by $\pi[k]$, is a finite play.

The total-payoff of a finite play $\pi = v_0 v_1 \cdots v_k$ is obtained by summing up the weights along $\pi$, i.e., $\mathbf{TP}(\pi) = \sum_{i=0}^{k-1} \omega(v_i, v_{i+1})$. In the following, we sometimes rely on the mean-payoff to obtain information about total-payoff objectives. The *mean-payoff* computes the average weight of $\pi$, i.e., if $k \geqslant 1$, $\mathbf{MP}(\pi) = \frac{1}{k} \sum_{i=0}^{k-1} \omega(v_i, v_{i+1})$, and $\mathbf{MP}(\pi) = 0$ when $k = 0$. These definitions are lifted to infinite plays as follows. The total-payoff of a play $\pi$ is given by $\mathbf{TP}(\pi) = \liminf_{k \to \infty} \mathbf{TP}(\pi[k])$.[2] Similarly, the mean-payoff of a play $\pi$ is given by $\mathbf{MP}(\pi) = \liminf_{k \to \infty} \mathbf{MP}(\pi[k])$. A weighted graph equipped with these payoffs is called a *total-payoff game* or a *mean-payoff game*, respectively.

A *strategy* for $\mathsf{Max}$ (respectively, $\mathsf{Min}$) in a game $\mathcal{G} = \langle V, E, \omega, \mathbf{P} \rangle$ (with $\mathbf{P}$ one of the previous payoffs), is a mapping $\sigma \colon V^* V_{\mathsf{Max}} \to V$ (respectively, $\sigma \colon V^* V_{\mathsf{Min}} \to V$) such that for all sequences $\pi = v_0 \cdots v_k$ with $v_k \in V_{\mathsf{Max}}$ (respectively, $v_k \in V_{\mathsf{Min}}$), $(v_k, \sigma(\pi)) \in E$. A play or finite play $\pi = v_0 v_1 \cdots$ conforms to a strategy $\sigma$ of $\mathsf{Max}$ (respectively, $\mathsf{Min}$) if for all $k$ such that $v_k \in V_{\mathsf{Max}}$ (respectively, $v_k \in V_{\mathsf{Min}}$), $v_{k+1} = \sigma(\pi[k])$. A strategy $\sigma$ is *memoryless*

[1] An example of practical application would be to perform controller synthesis taking into account energy consumption. On the other hand, the problem of computing the values in certain classes of priced timed games has recently been reduced to computing the values in min-cost reachability games [3].

[2] Our results can easily be extended by substituting a lim sup for the lim inf. The lim inf is more natural since we adopt the point of view of the maximiser $\mathsf{Max}$, hence the lim inf is the *worst* partial sum seen infinitely often.

if for all finite plays $\pi, \pi'$, we have $\sigma(\pi v) = \sigma(\pi' v)$ for all $v$. A strategy $\sigma$ is said to be *finite-memory* if it can be encoded in a deterministic Moore machine, $\langle M, m_0, \mathsf{up}, \mathsf{dec}\rangle$, where $M$ is a finite set representing the memory of the strategy, with an initial memory content $m_0 \in M$, $\mathsf{up}\colon M \times V \to M$ is a memory-update function, and $\mathsf{dec}\colon M \times V \to V$ a decision function such that for every finite play $\pi$ and vertex $v$, $\sigma(\pi v) = \mathsf{dec}(\mathsf{mem}(\pi v), v)$ where $\mathsf{mem}(\pi)$ is defined by induction on the length of the finite play $\pi$ as follows: $\mathsf{mem}(v_0) = m_0$, and $\mathsf{mem}(\pi v) = \mathsf{up}(\mathsf{mem}(\pi), v)$. We say that $|M|$ is the *size* of the strategy.

For all strategies $\sigma_{\mathsf{Max}}$ and $\sigma_{\mathsf{Min}}$, for all vertices $v$, we let $\mathsf{Play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}})$ be the outcome of $\sigma_{\mathsf{Max}}$ and $\sigma_{\mathsf{Min}}$, defined as the unique play conforming to $\sigma_{\mathsf{Max}}$ and $\sigma_{\mathsf{Min}}$ and starting in $v$. Naturally, the objective of $\mathsf{Max}$ is to maximise its payoff. In this model of zero-sum game, $\mathsf{Min}$ then wants to minimise the payoff of $\mathsf{Max}$. Formally, we let $\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Max}})$ and $\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Min}})$ be the respective values of the strategies, defined as (recall that $\mathbf{P}$ is either $\mathbf{TP}$ or $\mathbf{MP}$): $\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Max}}) = \inf_{\sigma_{\mathsf{Min}}} \mathbf{P}(\mathsf{Play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}}))$ and $\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Min}}) = \sup_{\sigma_{\mathsf{Max}}} \mathbf{P}(\mathsf{Play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}}))$. Finally, for all vertices $v$, we let $\underline{\mathsf{Val}}_{\mathcal{G}}(v) = \sup_{\sigma_{\mathsf{Max}}} \mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Max}})$ and $\overline{\mathsf{Val}}_{\mathcal{G}}(v) = \inf_{\sigma_{\mathsf{Min}}} \mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Min}})$ be the *lower* and *upper values* of $v$ respectively. We may easily show that $\underline{\mathsf{Val}}_{\mathcal{G}} \preccurlyeq \overline{\mathsf{Val}}_{\mathcal{G}}$. We say that strategies $\sigma^*_{\mathsf{Max}}$ of $\mathsf{Max}$ and $\sigma^*_{\mathsf{Min}}$ of $\mathsf{Min}$ are optimal if, for all vertices $v$: $\mathsf{Val}_{\mathcal{G}}(v, \sigma^*_{\mathsf{Max}}) = \underline{\mathsf{Val}}_{\mathcal{G}}(v)$ and $\mathsf{Val}_{\mathcal{G}}(v, \sigma^*_{\mathsf{Min}}) = \overline{\mathsf{Val}}_{\mathcal{G}}(v)$ respectively. We say that a game $\mathcal{G}$ is *determined* if for all vertices $v$, its lower and upper values are equal. In that case, we write $\mathsf{Val}_{\mathcal{G}}(v) = \underline{\mathsf{Val}}_{\mathcal{G}}(v) = \overline{\mathsf{Val}}_{\mathcal{G}}(v)$, and refer to it as the *value* of $v$. If the game is clear from the context, we may drop the index $\mathcal{G}$ of all previous values. Mean-payoff and total-payoff games are known to be determined, with the existence of optimal memoryless strategies [17, 10].

Total-payoff games have been mainly considered as a refinement of mean-payoff games [10]. Indeed, if the mean-payoff value of a game is positive (respectively, negative), its total-payoff value is necessarily $+\infty$ (respectively, $-\infty$). When the mean-payoff value is 0 however, the total-payoff is necessarily different from $+\infty$ and $-\infty$, hence total-payoff games are particularly useful in this case. Deciding whether the total-payoff value of a vertex is positive can be achieved in NP $\cap$ co-NP. In [9], the complexity is refined to UP $\cap$ co-UP, and values are shown to be effectively computable solving nested fixed point equations with a strategy iteration algorithm working in exponential time in the worst case.

Our aim is to give a pseudo-polynomial algorithm solving total-payoff games. In many cases, (e.g., mean-payoff games), a successful way to obtain such an efficient algorithm is the *value iteration paradigm*. Intuitively, value iteration algorithms compute successive approximations $x_0, x_1, \ldots, x_i, \ldots$ of the game value by restricting the number of turns that the players are allowed to play: $x_i$ is the vector of optimal values achievable when the players play at most $i$ turns. The sequence of values is computed by means of an operator $\mathcal{F}$, letting $v_{i+1} = \mathcal{F}(v_i)$ for all $i$. Good properties (Scott-continuity and monotonicity) of $\mathcal{F}$ ensure convergence towards its smallest or greatest fixed point (depending on the value of $x_0$), which, in some cases, happens to be the value of the game. Let us briefly explain why such a simple approach fails with total-payoff games. In our case, the operator $\mathcal{F}$ is such that $\mathcal{F}(x)(v) = \max_{v' \in E(v)} \omega(v, v') + x(v')$ for all $v \in V_{\mathsf{Max}}$ and $\mathcal{F}(x)(v) = \min_{v' \in E(v)} \omega(v, v') + x(v')$ for all $v \in V_{\mathsf{Min}}$. This definition matches the intuition that $x_i$ are optimal values after $i$ turns.

Then, consider the example of Figure 1(a), limited to vertices $\{v_3, v_4, v_5\}$ for simplicity. Observe that there are two simple cycles with weight 0, hence the total-payoff value of this game is finite. $\mathsf{Max}$ has the choice between cycling into one of these two cycles. It is easy to check that $\mathsf{Max}$'s optimal choice is to enforce the cycle between $v_4$ and $v_5$, securing a payoff of $-1$ from $v_4$ (because of the lim inf definition of $\mathbf{TP}$). Hence, the values of $x_3$, $x_4$ and $x_5$ are respectively 1, $-1$ and 0. In this game, we have $\mathcal{F}(x_3, x_4, x_5) =$

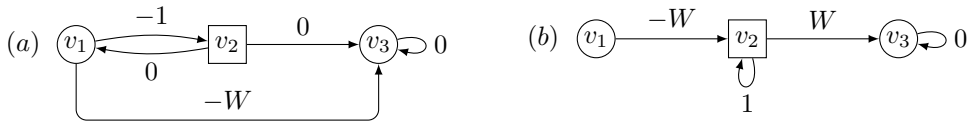


**Figure 2** Two weighted graphs.

$\big(2+x_4, \max(-2+x_3, -1+x_5), 1+x_4\big)$, and the vector $(1,-1,0)$ is indeed a fixed point of $\mathcal{F}$. However, it is neither the greatest nor the smallest fixed point of $\mathcal{F}$, since *if* $x$ is a fixed point of $\mathcal{F}$, *then* $x+(a,a,a)$ is also a fixed point, for all constant $a \in \mathbb{Z}$. If we try to initialise the value iteration algorithm with value $(0,0,0)$, which could seem a reasonable choice, the sequence of computed vectors is: $(0,0,0)$, $(2,-1,1)$, $(1,0,0)$, $(2,-1,1)$, $(1,0,0)$, ... that is not stationary, and does not even contain $(1,-1,0)$. Thus, it seems difficult to compute the actual game values with an iterative algorithm relying on the $\mathcal{F}$ operator, as in the case of mean-payoff games.[3] Notice that, in the previous example, the Zwick and Paterson's algorithm [17] to solve mean-payoff games would easily conclude from the sequence above, since the vectors of interest are then the one divided by the length of the current sequence, i.e., $(0,0,0)$, $(1,-0.5,0.5)$, $(0.33,0,0)$, $(0.5,-0.25,0.25)$, $(0.2,0,0)$, ... indeed converging towards $(0,0,0)$, the mean-payoff values of this game.

Instead, as explained in the introduction, we propose a different approach that consists in reducing total-payoff games to min-cost reachability games where Min must enforce a reachability objective on top of his optimisation objective. The aim of the next section is to study these games, and we reduce total-payoff games to them in Section 4.

## 3 Min-cost reachability games

In this section, we consider *min-cost reachability games* (MCR games for short), a variant of total-payoff games where one player has a reachability objective that he must fulfil first, before optimising his quantitative objective. Without loss of generality, we assign the reachability objective to player Min, as this will make our reduction from total-payoff games easier to explain. Hence, when the target is not reached along a path, its payoff shall be the worst possible for Min, i.e., $+\infty$. Formally, an MCR game is played on a weighted graph $\langle V, E, \omega\rangle$ equipped with a target set of vertices $T \subseteq V$. The payoff $T$-$\mathbf{MCR}(\pi)$ of a play $\pi = v_0 v_1 \dots$ is given by $T$-$\mathbf{MCR}(\pi) = +\infty$ if the play avoids $T$, i.e., if for all $k \geqslant 0$, $v_k \notin T$, and $T$-$\mathbf{MCR}(\pi) = \mathbf{TP}(\pi[k])$ if $k$ is the least position in $\pi$ such that $v_k \in T$. Lower and upper values are then defined as in Section 2. By an indirect consequence of Martin's theorem [12], we can show that MCR games are also determined. Optimal strategies may however not exist, as we will see later.

As an example, consider the MCR game played on the weighted graph of Figure 2(*a*), where $W$ is a positive integer and $v_3$ is the target.

We claim that the values of vertices $v_1$ and $v_2$ are both $-W$. Indeed, consider the following strategy for Min: during each of the first $W$ visits to $v_2$ (if any), go to $v_1$; else, go to $v_3$. Clearly, this strategy ensures that the target will eventually be reached, and that either (i) edge $(v_1, v_3)$ (with weight $-W$) will eventually be traversed; or (ii) edge $(v_1, v_2)$ (with weight $-1$) will be traversed at least $W$ times. Hence, in all plays following this strategy, the

[3] In the context of stochastic models like Markov decision processes, Strauch [14] already noticed that in the presence of arbitrary weights, the value iteration algorithm does not necessarily converge towards the accurate value: see [13, Ex. 7.3.3] for a detailed explanation.

payoff will be at most $-W$. This strategy allows Min to secure $-W$, but he cannot ensure a lower payoff, since Max always has the opportunity to take the edge $(v_1, v_3)$ (with weight $-W$) instead of cycling between $v_1$ and $v_2$. Hence, Max's optimal choice is to follow the edge $(v_1, v_3)$ as soon as $v_1$ is reached, securing a payoff of $-W$. The Min strategy we have just given is optimal, and there is *no optimal memoryless strategy* for Min. Indeed, always playing $(v_2, v_3)$ does not ensure a payoff $\leqslant -W$; and, always playing $(v_2, v_1)$ does not guarantee to reach the target, and this strategy has thus value $+\infty$.

Let us note that Björklund and Vorobyov introduce in [1] the *longest shortest path problem* (LSP for short) and propose a pseudo-polynomial time algorithm to solve it. However, their definition has several subtle but important differences to ours, such as definition of the payoff of a play (equivalently, the length of a path). As an example, in the game of Figure 2(a), the play $\pi = (v_1 v_2)^\omega$ (that never reaches the target) has length $-\infty$ in their setting, while, in our setting, $\{v_3\}$-$\mathbf{MCR}(\pi) = +\infty$. Moreover, even if a pre-treatment would hypothetically allow one to use the LSP algorithm to solve MCR games, our solution is simpler to implement with the same worst-case complexity and heuristics only applicable to our value iteration solution. We now present our contributions for MCR games:

▶ **Theorem 1.** *Let $\mathcal{G} = \langle V, E, \omega, T\text{-}\mathbf{MCR}\rangle$ be an MCR game.*

1. *For $v \in V$, deciding whether $\mathsf{Val}(v) = +\infty$ can be done in polynomial time.*
2. *For $v \in V$, deciding whether $\mathsf{Val}(v) = -\infty$ is as hard as mean-payoff, in* NP ∩ co-NP *and can be achieved in pseudo-polynomial time.*
3. *If $\mathsf{Val}(v) \neq -\infty$ for all vertices $v \in V$, then both players have optimal strategies. Moreover,* Max *always has a memoryless optimal strategy, while* Min *may require finite (pseudo-polynomial) memory in his optimal strategy.*
4. *Computing all values $\mathsf{Val}(v)$ (for $v \in V$), as well as optimal strategies (if they exist) for both players, can be done in (pseudo-polynomial) time $O(|V|^2|E|W)$.*

To prove the first item it suffices to notice that vertices with value $+\infty$ are exactly those from which Min cannot reach the target. Therefore the problem reduces to deciding the winner in a classical reachability game, that can be solved in polynomial time [16], using the classical *attractor* construction: in vertices of value $+\infty$, Min may play indifferently, while Max has an optimal memoryless strategy consisting in avoiding the attractor.

To prove the second item, it suffices first to notice that vertices with value $-\infty$ are exactly those with a value $< 0$ in the mean-payoff game played on the same graph. On the other hand, we can show that any mean-payoff game can be transformed (in polynomial time) into an MCR game such that a vertex has value $< 0$ in the mean-payoff game if and only if the value of its corresponding vertex in the MCR game is $-\infty$. The rest of this section focuses on the proof of the third and fourth items. We start by explaining how to compute the values in pseudo-polynomial, and we discuss optimal strategies afterward.

**Computing the values.** From now on, we assume, without loss of generality, that there is exactly one target vertex denoted by $\mathtt{t}$, and the only outgoing edge from $\mathtt{t}$ is a self loop with weight 0: this is reflected by denoting $\mathbf{MCR}$ the payoff mapping $\{\mathtt{t}\}$-$\mathbf{MCR}$. Our value iteration algorithm for MCR games is given in Algorithm 1. To establish its correctness, we rely mainly on the operator $\mathcal{F}$, which denotes the function $\mathbb{Z}_\infty^V \to \mathbb{Z}_\infty^V$ mapping every vector $x \in \mathbb{Z}_\infty^V$ to $\mathcal{F}(x)$ defined by $\mathcal{F}(x)(\mathtt{t}) = 0$ and

$$\mathcal{F}(x)(v) = \begin{cases} \max\limits_{v' \in E(v)} \big(\omega(v, v') + x(v')\big) & \text{if } v \in V_{\mathsf{Max}} \setminus \{\mathtt{t}\} \\ \min\limits_{v' \in E(v)} \big(\omega(v, v') + x(v')\big) & \text{if } v \in V_{\mathsf{Min}} \setminus \{\mathtt{t}\} \end{cases}$$

**Algorithm 1:** Value iteration for min-cost reachability games

**Input**: MCR game $\langle V, E, \omega, \mathbf{MCR}\rangle$, $W$ largest weight in absolute value

1 $\mathsf{X}(\mathtt{t}) := 0$
2 **foreach** $v \in V \setminus \{\mathtt{t}\}$ **do** $\mathsf{X}(v) := +\infty$
3 **repeat**
4 $\quad \mathsf{X}_{pre} := \mathsf{X}$
5 $\quad$ **foreach** $v \in V_{\mathsf{Max}} \setminus \{\mathtt{t}\}$ **do** $\mathsf{X}(v) := \max_{v' \in E(v)} \big(\omega(v,v') + \mathsf{X}_{pre}(v')\big)$
6 $\quad$ **foreach** $v \in V_{\mathsf{Min}} \setminus \{\mathtt{t}\}$ **do** $\mathsf{X}(v) := \min_{v' \in E(v)} \big(\omega(v,v') + \mathsf{X}_{pre}(v')\big)$
7 $\quad$ **foreach** $v \in V \setminus \{\mathtt{t}\}$ such that $\mathsf{X}(v) < -(|V|-1)W$ **do** $\mathsf{X}(v) := -\infty$
8 **until** $\mathsf{X} = \mathsf{X}_{pre}$
9 **return** $\mathsf{X}$

More precisely, we are interested in the sequence of iterates $x_i = \mathcal{F}(x_{i-1})$ of $\mathcal{F}$ from the initial vector $x_0$ defined by $x_0(v) = +\infty$ for all $v \neq \mathtt{t}$, and $x_0(\mathtt{t}) = 0$. The intuition behind the sequence $(x_i)_{i \geqslant 0}$ is that $x_i$ *is the value of the game if we impose that* Min *must reach the target within* $i$ *steps* (and get a payoff of $+\infty$ if he fails to do so). Formally, for a play $\pi = v_0 v_1 \cdots v_i \cdots$, we let $\mathbf{MCR}^{\leqslant i}(\pi) = \mathbf{MCR}(\pi)$ if $v_k = \mathtt{t}$ for some $k \leqslant i$, and $\mathbf{MCR}^{\leqslant i}(\pi) = +\infty$ otherwise. We further let $\overline{\mathsf{Val}}^{\leqslant i}(v) = \inf_{\sigma_{\mathsf{Min}}} \sup_{\sigma_{\mathsf{Max}}} \mathbf{MCR}^{\leqslant i}(\mathsf{Play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}}))$ (where $\sigma_{\mathsf{Max}}$ and $\sigma_{\mathsf{Min}}$ are respectively strategies of Max and Min). We can show that the operator $\mathcal{F}$ allows one to compute the sequence $(\overline{\mathsf{Val}}^{\leqslant i})_{i \geqslant 0}$, i.e., for all $i \geqslant 0$: $x_i = \overline{\mathsf{Val}}^{\leqslant i}$.

Let us first show that the algorithm is *correct* when the values of all nodes are *finite*. Thanks to this characterisation, and by definition of $\overline{\mathsf{Val}}^{\leqslant i}$, it is easy to see that, for all $i \geqslant 0$: $x_i = \overline{\mathsf{Val}}^{\leqslant i} \succcurlyeq \overline{\mathsf{Val}} = \mathsf{Val}$. Moreover, $\mathcal{F}$ is a monotonic operator over the complete lattice $\mathbb{Z}_\infty^V$. By Knaster-Tarski's theorem, the fixed points of $\mathcal{F}$ form a complete lattice and $\mathcal{F}$ admits a greatest fixed point. By Kleene's fixed point theorem, using the Scott-continuity of $\mathcal{F}$, this greatest fixed point can be obtained as the limit of the non-increasing sequence of iterates $(\mathcal{F}^i(\overline{x}))_{i \geqslant 0}$ starting in the maximal vector $\overline{x}$ defined by $\overline{x}(v) = +\infty$ for all $v \in V$. As $x_0 = \mathcal{F}(\overline{x})$, the sequence $(x_i)_{i \geqslant 0}$ is also non-increasing (i.e., $x_i \succcurlyeq x_{i+1}$, for all $i \geqslant 0$) and converges towards the greatest fixed point of $\mathcal{F}$. We can further show that the value of the game Val is actually the greatest fixed point of $\mathcal{F}$. Moreover, we can bound the number of steps needed to reach that fixed point (when all values are finite – this is the point where this hypothesis is crucial), by carefully observing the possible vectors that can be computed by the algorithm: the sequence $(x_i)_{i \geqslant 0}$ is non-increasing, and stabilises after at most $(2|V|-1)W|V| + |V|$ steps on Val.

Thus, computing the sequence $(x_i)_{i \geqslant 0}$ up to stabilisation yields the values of all vertices in an MCR game *if all values are finite*. Were it not for line 7, Algorithm 1 would compute exactly this sequence. We claim that Algorithm 1 is correct even when vertices have values in $\{-\infty, +\infty\}$. Line 7 allows to cope with vertices whose value is $-\infty$: when the algorithm detects that Min can secure a value small enough from a vertex $v$, it sets $v$'s value to $-\infty$. Intuitively, this is correct because if Min can guarantee a payoff smaller than $-(|V|-1) \times W$, he can force a negative cycle from which he can reach $\mathtt{t}$ with an arbitrarily small value. Hence, one can ensure that, after $i$ iterations of the loop, $x_{i-1} \succcurlyeq \mathsf{X} \succcurlyeq \mathsf{Val}$, and the sequence still converges to Val, the greatest fixed point of $\mathcal{F}$. Finally, if some vertex $v$ has value $+\infty$, one can check that $\mathsf{X}(v) = +\infty$ is an invariant of the loop. From that point, one can prove the correctness of the algorithm. Thus, the algorithm executes $O(|V|^2 W)$ iterations. Since each iteration can be performed in $O(|E|)$, the algorithm has a complexity of $O(|V|^2|E|W)$,

as announced in Theorem 1. As an example, consider the min-cost reachability game of Figure 2($a$). The successive values for vertices $(v_1, v_2)$ (value of the target $v_3$ is always 0) computed by the value iteration algorithm are the following: $(+\infty, +\infty)$, $(+\infty, 0)$, $(-1, 0)$, $(-1, -1)$, $(-2, -1)$, $(-2, -2), \ldots, (-W, -W+1)$, $(-W, -W)$. This requires $2W$ steps to converge (hence a pseudo-polynomial time).

**Computing optimal strategies for both players.** We now turn to the proof of the third item of Theorem 1, supposing that every vertex $v$ of the game has a finite value $\mathsf{Val}(v) \in \mathbb{Z}$ (the case where $\mathsf{Val}(v) = +\infty$ is delt with the attractor construction).

Observe first that, Min may *need memory* to play optimally, as already shown by the example in Figure 2($a$), where the target is $v_3$. Nevertheless, let us briefly explain why optimal strategies for Min always exist, with a memory of pseudo-polynomial size. We extract from the sequence $(x_i)_{i \geqslant 0}$ defined above (or equivalently, from the sequence of vectors $\mathsf{X}$ of Algorithm 1) the optimal strategy $\sigma^*_{\mathsf{Min}}$ as follows. Let $k$ be the first index such that $x_{k+1} = x_k$. Then, for every play $\pi$ ending in vertex $v \in V_{\mathsf{Min}}$, we let $\sigma^*_{\mathsf{Min}}(\pi) = \operatorname{argmin}_{v' \in E(v)} \big(\omega(v, v') + x_{k-|\pi|-1}(v')\big)$, if $|\pi| < k$, and $\sigma^*_{\mathsf{Min}}(\pi) = \operatorname{argmin}_{v' \in E(v)} \big(\omega(v, v') + x_0(v')\big)$ otherwise (those argmin may not be unique, but we can indifferently pick any of them). Since $\sigma_{\mathsf{Min}}$ only requires to know the last vertex and the length of the prefix up to $k$, and since $k \leqslant (2|V|-1)W|V|+|V|$ as explained above, $\sigma^*_{\mathsf{Min}}$ needs a memory of pseudo-polynomial size only. Moreover, it can be computed with the sequence of vectors $\mathsf{X}$ in Algorithm 1. It is not difficult to verify by induction that this strategy is optimal for Min. While optimal, this strategy might not be practical, for instance, in the framework of controller synthesis. Implementing it would require to store the full sequence $(x_i)_{i \geqslant 0}$ up to convergence step $k$ (possibly pseudo-polynomial) in a table, and to query this large table each time the strategy is called. Instead, an alternative optimal strategy $\sigma'_{\mathsf{Min}}$ can be construct, that consists in playing successively two *memoryless strategies* $\sigma^1_{\mathsf{Min}}$ and $\sigma^2_{\mathsf{Min}}$ ($\sigma^2_{\mathsf{Min}}$ being given by the attractor construction). To determine when to switch from $\sigma^1_{\mathsf{Min}}$ to $\sigma^2_{\mathsf{Min}}$, $\sigma'_{\mathsf{Min}}$ maintains a counter that is stored in a *polynomial* number of bits, thus the memory footprints of $\sigma'_{\mathsf{Min}}$ and $\sigma^*_{\mathsf{Min}}$ are comparable. However, $\sigma'_{\mathsf{Min}}$ is easier to implement, because $\sigma^1_{\mathsf{Min}}$ and $\sigma^2_{\mathsf{Min}}$ can be described by a pair of tables of linear size, and, apart from querying those tables, $\sigma'_{\mathsf{Min}}$ consists only in incrementing and testing the counter to determine when to switch. Moreover, this succession of two memoryless strategies allows us to also get some interesting strategy in case of vertices with values $-\infty$: indeed, we can still compute this pair of strategies, and simply modify the switching policy to run for a sufficiently long time to guarantee a value less than a given threshold. In the following, we call such a strategy a *switching strategy*.

Finally, we can show that, contrary to Min, Max always has a *memoryless optimal strategy* $\sigma^*_{\mathsf{Max}}$ defined by $\sigma^*_{\mathsf{Max}}(\pi) = \operatorname{argmax}_{v' \in E(v)} (\omega(v, v') + \mathsf{Val}(v'))$ for all finite plays $\pi$ ending in $v \in V_{\mathsf{Max}}$. For example, in the game of Figure 2($a$), $\sigma^*_{\mathsf{Max}}(\pi v_2) = v_3$ for all $\pi$, since $\mathsf{Val}(v_3) = 0$ and $\mathsf{Val}(v_1) = -W$. Moreover, the previously described optimal strategies can be computed along the execution of Algorithm 1. Finally, we can show that, for all vertices $v$, the pair of optimal strategies we have just defined yields a play $\mathsf{Play}(v, \sigma^*_{\mathsf{Max}}, \sigma^*_{\mathsf{Min}})$ which is *non-looping*, i.e., never visits the same vertex twice before reaching the target. For instance, still in the game of Figure 2($a$), $\mathsf{Play}(v_1, \sigma^*_{\mathsf{Max}}, \sigma^*_{\mathsf{Min}}) = v_1 v_2 v_3^\omega$.

## 4 An efficient algorithm to solve total-payoff games

We now turn our attention back to total-payoff games (without reachability objective), and discuss our main contribution. Building on the results of the previous section, we introduce

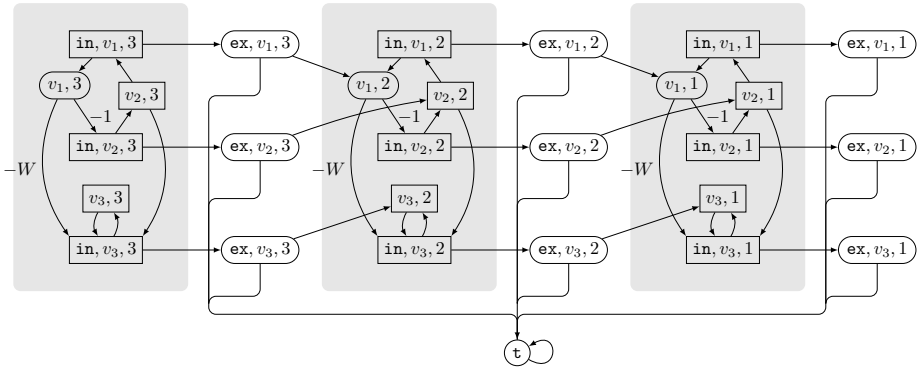


**Figure 3** MCR game $\mathcal{G}^3$ associated with the total-payoff game of Figure 2($a$).

the *first* (as far as we know) *pseudo-polynomial time algorithm* for solving those games in the presence of arbitrary weights, thanks to a reduction from total-payoff games to min-cost reachability games. The MCR game produced by the reduction has size pseudo-polynomial in the size of the original total-payoff game. Then, we show how to compute the values of the total-payoff game without building the entire MCR game, and explain how to deduce memoryless optimal strategies from the computation of our algorithm.

**Reduction to min-cost reachability games.** We provide a transformation from a total-payoff game $\mathcal{G} = \langle V, E, \omega, \mathbf{TP}\rangle$ to a min-cost reachability game $\mathcal{G}^K$ such that the values of $\mathcal{G}$ can be extracted from the values in $\mathcal{G}^K$ (as formalised below). Intuitively, $\mathcal{G}^K$ simulates the game where players play in $\mathcal{G}$; $\mathsf{Min}$ may propose to stop playing and reach a fresh vertex $\mathtt{t}$ acting as the target; $\mathsf{Max}$ can then accept, in which case we reach the target, or refuse at most $K$ times, in which case the game continues. Structurally, $\mathcal{G}^K$ consists of a sequence of copies of $\mathcal{G}$ along with some new states that we now describe formally. We let $\mathtt{t}$ be a fresh vertex, and, for all $n \geqslant 1$, we define the min-cost reachability game $\mathcal{G}^n = \langle V^n, E^n, \omega^n, \{\mathtt{t}\}\text{-}\mathbf{MCR}\rangle$ where $V^n_{\mathsf{Max}}$ (respectively, $V^n_{\mathsf{Min}}$) consists of $n$ copies $(v, j)$, with $1 \leqslant j \leqslant n$, of each vertex $v \in V_{\mathsf{Max}}$ (respectively, $v \in V_{\mathsf{Min}}$) and some *exterior vertices* $(\mathtt{ex}, v, j)$ for all $v \in V$ and $1 \leqslant j \leqslant n$ (respectively, *interior vertices* $(\mathtt{in}, v, j)$ for all $v \in V$ and $1 \leqslant j \leqslant n$). Moreover, $V^n_{\mathsf{Max}}$ contains the fresh target vertex $\mathtt{t}$. Edges are given by

$$\begin{aligned} E^n = {} & \{(\mathtt{t},\mathtt{t})\} \uplus \big\{\big((v,j),(\mathtt{in},v',j)\big) \mid (v,v') \in E, 1 \leqslant j \leqslant n\big\} \\ & \uplus \big\{\big((\mathtt{in},v,j),(v,j)\big) \mid v \in V, 1 \leqslant j \leqslant n\big\} \uplus \big\{\big((\mathtt{ex},v,j),\mathtt{t}\big) \mid v \in V, 1 \leqslant j \leqslant n\big\} \\ & \uplus \big\{\big((\mathtt{in},v,j),(\mathtt{ex},v,j)\big) \mid v \in V, 1 \leqslant j \leqslant n\big\} \\ & \uplus \big\{\big((\mathtt{ex},v,j),(v,j-1)\big) \mid v \in V, 1 < j \leqslant n\big\}. \end{aligned}$$

All edge weights are zero, except edges $\big((v,j),(\mathtt{in},v',j)\big)$ that have weight $\omega(v,v')$.

For example, considering the weighted graph of Figure 2($a$), the corresponding reachability total-payoff game $\mathcal{G}^3$ is depicted in Figure 3 (where weights 0 have been removed). The next proposition formalises the relationship between the two games.

▶ **Proposition 2.** *Let $K = |V|(2(|V|-1)W+1)$. For all $v \in V$ and $k \geqslant K$,*

- $\mathsf{Val}_{\mathcal{G}}(v) \neq +\infty$ *if and only if* $\mathsf{Val}_{\mathcal{G}}(v) = \mathsf{Val}_{\mathcal{G}^k}((v,k))$*;*
- $\mathsf{Val}_{\mathcal{G}}(v) = +\infty$ *if and only if* $\mathsf{Val}_{\mathcal{G}^k}((v,k)) \geqslant (|V|-1)W+1$.

The bound $K$ is found by using the fact (informally described in the previous section) that if not infinite, the value of a min-cost reachability game belongs in $[-(|V|-1)\times W+1, |V|\times W]$,

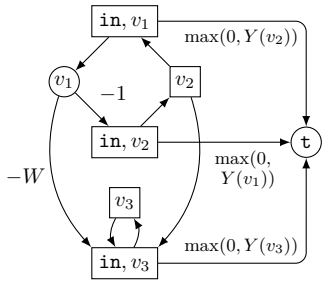


**Figure 4** MCR game $\mathcal{G}_Y$ associated with the total-payoff game of Figure 2($a$).

and that after enough visits of the same vertex, an adequate loop ensures that $\mathcal{G}^k$ verifies the above properties.

**Value iteration algorithm for total-payoff games.** By Proposition 2, an immediate way to obtain a value iteration algorithm for total-payoff games is to build game $\mathcal{G}^K$, run Algorithm 1 on it, and map the computed values back to $\mathcal{G}$. We take advantage of the structure of $\mathcal{G}^K$ to provide a better algorithm that avoids building $\mathcal{G}^K$. We first compute the values of the vertices in the *last copy of the game* (vertices of the form $(v, 1)$, $(\texttt{in}, v, 1)$ and $(\texttt{ex}, v, 1)$), then of those in the penultimate (vertices of the form $(v, 2)$, $(\texttt{in}, v, 2)$ and $(\texttt{ex}, v, 2)$), and so on.

We formalise this idea as follows. Let $Z^j$ be a vector mapping each vertex $v$ of $\mathcal{G}$ to the value $Z^j(v)$ of vertex $(v, j)$ in $\mathcal{G}^K$. Then, let us define an operator $\mathcal{H}$ such that $Z^{j+1} = \mathcal{H}(Z^j)$. The intuition behind the definition of $\mathcal{H}(Y)$ for some vector $Y$, is to extract from $\mathcal{G}^K$ one copy of the game, and make $Y$ appear in the weights of some edges as illustrated in Figure 4. This game, $\mathcal{G}_Y$, simulates a play in $\mathcal{G}$ in which $\mathsf{Min}$ can opt for 'leaving the game' at each round (by moving to the target), obtaining $\max(0, Y(v))$, if $v$ is the current vertex. Then $\mathcal{H}(Y)(v)$ is defined as the value of $v$ in $\mathcal{G}_Y$. By construction, it is easy to see that $Z^{j+1} = \mathcal{H}(Z^j)$ holds for all $j \geqslant 1$. Furthermore, we define $Z^0(v) = -\infty$ for all $v$, and have $Z^1 = \mathcal{H}(Z^0)$. One can prove the following properties of $\mathcal{H}$: (i) $\mathcal{H}$ is monotonic, but may not be Scott-continuous; (ii) the sequence $(Z^j)_{j \geqslant 0}$ converges towards $\mathsf{Val}_{\mathcal{G}}$.

We are now ready to introduce Algorithm 2 to solve total-payoff games. Intuitively, the outer loop computes, in variable $\mathsf{Y}$, a non-decreasing sequence of vectors whose limit is $\mathsf{Val}_{\mathcal{G}}$, and that is stationary (this is not necessarily the case for the sequence $(Z^j)_{j \geqslant 0}$). Line 1 initialises $\mathsf{Y}$ to $Z^0$. Each iteration of the outer loop amounts to running Algorithm 1 to compute $\mathcal{H}(\mathsf{Y}_{pre})$ (lines 3 to 10), then detecting if some vertices have value $+\infty$, updating $\mathsf{Y}$ accordingly (line 11, following the second item of Proposition 2). One can show that, for all $j > 0$, if we let $Y^j$ be the value of $\mathsf{Y}$ after the $j$-th iteration of the main loop, $Z^j \preccurlyeq Y^j \preccurlyeq \mathsf{Val}_{\mathcal{G}}$, which ensures the correctness of the algorithm.

▶ **Theorem 3.** *If a total-payoff game $\mathcal{G} = \langle V, E, \omega, \mathbf{TP} \rangle$ is given as input, Algorithm 2 outputs the vector $\mathsf{Val}_{\mathcal{G}}$ of optimal values, after at most $K = |V|(2(|V| - 1)W + 1)$ iterations of the external loop. The complexity of the algorithm is $O(|V|^4|E|W^2)$.*

The number of iterations in each internal loop is controlled by Theorem 1. On the example of Figure 2($a$), only 2 external iterations are necessary, but the number of iterations of each internal loop would be $2W$. By contrast, for the total-payoff game depicted in Figure 2($b$), each internal loop requires 2 iterations to converge, but the external loop takes $W$ iterations to stabilise. A combination of both examples would experience a pseudo-polynomial number of iterations to converge in both the internal and external loops, matching the $W^2$ term of the above complexity.

**Algorithm 2:** A value iteration algorithm for total-payoff games.

**Input**: Total-payoff game $\mathcal{G} = \langle V, E, \omega, \mathbf{TP}\rangle$, $W$ largest weight in absolute value

**foreach** $v \in V$ **do** $\mathsf{Y}(v) := -\infty$
**repeat**
  **foreach** $v \in V$ **do** $\mathsf{Y}_{pre}(v) := \mathsf{Y}(v)$; $\mathsf{Y}(v) := \max(0, \mathsf{Y}(v))$; $\mathsf{X}(v) := +\infty$
  **repeat**
    $\mathsf{X}_{pre} := \mathsf{X}$
    **foreach** $v \in V_{\mathsf{Max}}$ **do** $\mathsf{X}(v) := \max_{v' \in E(v)} \left[\omega(v, v') + \min(\mathsf{X}_{pre}(v'), \mathsf{Y}(v'))\right]$
    **foreach** $v \in V_{\mathsf{Min}}$ **do** $\mathsf{X}(v) := \min_{v' \in E(v)} \left[\omega(v, v') + \min(\mathsf{X}_{pre}(v'), \mathsf{Y}(v'))\right]$
    **foreach** $v \in V$ such that $\mathsf{X}(v) < -(|V| - 1)W$ **do** $\mathsf{X}(v) := -\infty$
  **until** $\mathsf{X} = \mathsf{X}_{pre}$
  $\mathsf{Y} := \mathsf{X}$
  **foreach** $v \in V$ such that $\mathsf{Y}(v) > (|V| - 1)W$ **do** $\mathsf{Y}(v) := +\infty$
**until** $\mathsf{Y} = \mathsf{Y}_{pre}$
**return** $\mathsf{Y}$

**Optimal strategies.** In Section 3, we have shown, for any min-cost reachability game, the existence of a pair of memoryless strategies permitting to reconstruct a *switching* optimal strategy for $\mathsf{Min}$ (if every vertex has value different from $-\infty$, or a strategy ensuring any possible threshold for vertices with value $-\infty$). If we apply this construction to the game $\mathcal{G}_{\mathsf{Val}_\mathcal{G}}$, we obtain a pair $(\sigma^1_{\mathsf{Min}}, \sigma^2_{\mathsf{Min}})$ of strategies (remember that $\sigma^2_{\mathsf{Min}}$ is a strategy obtained by the attractor construction, hence it will not be useful for us for total-payoff games). Consider the strategy $\overline{\sigma_{\mathsf{Min}}}$, obtained by projecting $\sigma^1_{\mathsf{Min}}$ on $V$ as follows: for all finite plays $\pi$ and vertex $v \in V_{\mathsf{Min}}$, let $\overline{\sigma_{\mathsf{Min}}}(\pi v) = v'$ if $\sigma^1_{\mathsf{Min}}(v) = (\mathtt{in}, v')$. We can show that $\overline{\sigma_{\mathsf{Min}}}$ is optimal for $\mathsf{Min}$ in $\mathcal{G}$. Notice that $\sigma^1_{\mathsf{Min}}$, and hence $\overline{\sigma_{\mathsf{Min}}}$, can be computed during the last iteration of the value iteration algorithm, as explained in the case of min-cost reachability. A similar construction can be done to compute an optimal strategy of $\mathsf{Max}$.

## 5 Implementation and heuristics

In this section, we report on a prototype implementation of our algorithms.[4] For convenience reasons, we have implemented them as an add-on to PRISM-games [5], although we could have chosen to extend another model-checker as we do not rely on the probabilistic features of PRISM models (i.e., we use the PRISM syntax of *stochastic multi-player games*, allowing arbitrary rewards, and forbidding probability distributions different of Dirac ones). We then use rPATL specifications of the form $\langle\langle C\rangle\rangle \mathsf{R}^{\min\,/\,\max=?}[\mathsf{F}^\infty \varphi]$ and $\langle\langle C\rangle\rangle \mathsf{R}^{\min\,/\,\max=?}[\mathsf{F}^c \bot]$ to model respectively min-cost reachability games and total-payoff games, where $C$ represents a coalition of players that want to minimise/maximise the payoff, and $\varphi$ is another rPATL formula describing the target set of vertices (for total-payoff games, such a formula is not necessary). We have tested our implementation on toy examples. On the parametric one studied after Theorem 3, obtained by mixing the graphs of Figure 2 and repeating them for $n$ layers, results obtained by applying our algorithm for total-payoff games are summarised in Table 1, where for each pair $(W, n)$, we give the time $t$ in seconds, the number $k_e$ of iterations in the external loop, and the total number $k_i$ of iterations in the internal loop.

[4] Source and binary files, as well as some examples, can be downloaded from `http://www.ulb.ac.be/di/verif/monmege/tool/TP-MCR/`.

**Table 1** Results of value iteration on a parametric example.

| | | without heuristics | | | with heuristics | | |
|---|---|---|---|---|---|---|---|
| $W$ | $n$ | $t$ | $k_e$ | $k_i$ | $t$ | $k_e$ | $k_i$ |
| 50 | 100 | 0.52s | 151 | 12,603 | 0.01s | 402 | 1,404 |
| 50 | 500 | 9.83s | 551 | 53,003 | 0.42s | 2,002 | 7,004 |
| 200 | 100 | 2.96s | 301 | 80,103 | 0.02s | 402 | 1,404 |
| 200 | 500 | 45.64s | 701 | 240,503 | 0.47s | 2,002 | 7,004 |
| 500 | 1,000 | 536s | 1,501 | 1,251,003 | 2.37s | 4,002 | 14,004 |

We close this section by sketching two techniques that can be used to speed up the computation of the fixed point in Algorithms 1 and 2. We fix a weighted graph $\langle V, E, \omega\rangle$. Both accelerations rely on a topological order of the strongly connected components (SCC for short) of the graph, given as a function $\mathsf{c}\colon V \to \mathbb{N}$, mapping each vertex to its *component*, verifying that (i) $\mathsf{c}(V) = \{0, \dots, p\}$ for some $p \geqslant 0$, (ii) $\mathsf{c}^{-1}(q)$ is a maximal SCC for all $q$, (iii) and $\mathsf{c}(v) \geqslant \mathsf{c}(v')$ for all $(v, v') \in E$.[5] In case of an MRC game with $\mathtt{t}$ the unique target, $\mathsf{c}^{-1}(0) = \{\mathtt{t}\}$. Intuitively, $\mathsf{c}$ induces a directed acyclic graph whose vertices are the sets $\mathsf{c}^{-1}(q)$ for all $q \in \mathsf{c}(V)$, and with an edge $(S_1, S_2)$ if and only if there are $v_1 \in S_1, v_2 \in S_2$ such that $(v_1, v_2) \in E$.

The *first acceleration heuristic* is a divide-and-conquer technique that consists in applying Algorithm 1 (or the inner loop of Algorithm 2) iteratively on each $\mathsf{c}^{-1}(q)$ for $q = 0, 1, 2, \dots, p$, using at each step the information computed during steps $j < q$ (since the value of a vertex $v$ depends only on the values of the vertices $v'$ such that $\mathsf{c}(v') \leqslant \mathsf{c}(v)$). The *second acceleration heuristic* consists in studying more precisely each component $\mathsf{c}^{-1}(q)$. Having already computed the optimal values $\mathsf{Val}(v)$ of vertices $v \in \mathsf{c}^{-1}(\{0, \dots, q-1\})$, we ask an oracle to precompute a finite set $S_v \subseteq \mathbb{Z}_\infty$ of possible optimal values for each vertex $v \in \mathsf{c}^{-1}(q)$. For MCR games and the inner iteration of the algorithm for total-payoff games, one way to construct such a set $S_v$ is to consider that possible optimal values are the one of non-looping paths inside the component exiting it, since, in MCR games, there exist optimal strategies for both players whose outcome is a non-looping path (see Section 3).

We can identify classes of weighted graphs for which there exists an oracle that runs in polynomial time and returns, for all vertices $v$, a set $S_v$ of polynomial size. On such classes, Algorithms 1 and 2, enhanced with our two acceleration techniques, *run in polynomial time*. For instance, for all fixed positive integers $L$, the class of weighted graphs where every component $\mathsf{c}^{-1}(q)$ uses at most $L$ distinct weights (that can be arbitrarily large in absolute value) satisfies this criterion. Table 1 contains the results obtained with the heuristics on the parametric example presented before. Observe that the acceleration technique permits here to decrease drastically the execution time, the number of iterations in both loops depending not even anymore on $W$. Even though the number of iterations in the external loop increases with heuristics, due to the decomposition, less computation is required in each internal loop since we only apply the computation for the active component.

## References

1 Henrik Björklund and Sergei Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155:210–229, 2007.

[5] Such a mapping is computable in linear time, e.g., by Tarjan's algorithm [15].

2 Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. To reach or not to reach? Efficient algorithms for total-payoff games. Research Report 1407.5030, arXiv, July 2014.

3 Thomas Brihaye, Gilles Geeraerts, Shankara Narayanan Krishna, Lakshmi Manasa, Benjamin Monmege, and Ashutosh Trivedi. Adding negative prices to priced timed games. In *Proceedings of the 25th International Conference on Concurrency Theory (CONCUR'14)*, volume 8704 of *Lecture Notes in Computer Science*, pages 560–575. Springer, 2014.

4 Luboš Brim, Jakub Chaloupka, Laurent Doyen, Rafaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods for System Design*, 38(2):97–118, 2011.

5 Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, David Parker, and Aistis Simaitis. Automatic verification of competitive stochastic systems. *Formal Methods in System Design*, 43(1):61–92, 2013.

6 Carlo Comin and Romeo Rizzi. An improved pseudo-polynomial upper bound for the value problem and optimal strategy synthesis in mean payoff games. Technical Report 1503.04426, arXiv, 2015.

7 Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8(2):109–113, 1979.

8 Emmanuel Filiot, Rafaella Gentilini, and Jean-François Raskin. Quantitative languages defined by functional automata. In *Proceedings of the 23rd International Conference on Concurrency theory (CONCUR'12)*, volume 7454 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2012.

9 Thomas Martin Gawlitza and Helmut Seidl. Games through nested fixpoints. In *Proceedings of the 21st International Conference on Computer Aided Verification (CAV'09)*, volume 5643 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2009.

10 Hugo Gimbert and Wiesław Zielonka. When can you play positionally? In *Proceedings of the 29th International Conference on Mathematical Foundations of Computer Science (MFCS'04)*, volume 3153 of *Lecture Notes in Computer Science*, pages 686–698. Springer, 2004.

11 Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir Gurvich, Gabor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43:204–233, 2008.

12 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

13 Martin L. Puterman. *Markov Decision Processes.* John Wiley & Sons, Inc., New York, NY, 1994.

14 Ralph E. Strauch. Negative dynamic programming. *The Annals of Mathematical Statistics*, 37:871–890, 1966.

15 Robert E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

16 Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1995.

17 Uri Zwick and Michael S. Paterson. The complexity of mean payoff games. *Theoretical Computer Science*, 158:343–359, 1996.