The complexity of mean payoff games on graphs *

Uri Zwick † *Mike Paterson* ‡

July 26, 1995

Abstract

We study the complexity of finding the values and optimal strategies of *mean payoff games* on graphs, a family of perfect information games introduced by Ehrenfeucht and Mycielski and considered by Gurvich, Karzanov and Khachiyan. We describe a pseudo-polynomial time algorithm for the solution of such games, the decision problem for which is in NP ∩ co-NP. Finally, we describe a polynomial reduction from mean payoff games to the *simple stochastic games* studied by Condon. These games are also known to be in NP ∩ co-NP, but no polynomial or pseudo-polynomial time algorithm is known for them.

1 Introduction

Let $G = (V, E)$ be a finite directed graph in which each vertex has at least one edge going out of it. Let $w : E \to \{-W, \ldots, 0, \ldots, W\}$ be a function that assigns an integral weight to each edge of G. Ehrenfeucht and Mycielski [EM79] studied the following infinite two-person game played on such a graph. The game starts at a vertex $a_0 \in V$. The first player chooses an edge $e_1 = (a_0, a_1) \in E$. The second player then chooses an edge $e_2 = (a_1, a_2) \in E$, and so on indefinitely. The first player wants to maximise $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(e_i)$. The second player wants to minimise $\limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(e_i)$. Ehrenfeucht and Mycielski show that each such game has a value ν such that the first player has a strategy that ensures that $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(e_i) \geq \nu$, while the second player has a strategy that ensures that $\limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(e_i) \leq \nu$. Furthermore, they show that both players can achieve this value using a *positional strategy*, i.e., a strategy in which the next move depends only on the vertex from which the player is to move.

As the players in these *mean payoff games* move alternatively, we may assume, without loss of generality, that the graph $G = (V, E)$ on which such a game is played is bipartite, with V_1 and V_2 being the partition of the vertices into the two 'sides' and with $E = E_1 \cup E_2$ such that $E_1 \subseteq V_1 \times V_2$ and $E_2 \subseteq V_2 \times V_1$. If the original graph is not bipartite, we simply duplicate the set of vertices.

To obtain their results for the infinite game, Ehrenfeucht and Mycielski [EM79] also consider the following finite version of the game. Again the game starts at a specific vertex of the graph $G = (V, E)$, which is assumed to be bipartite. The players alternate in choosing successive edges that form a path, but the game ends as soon as a cycle is formed. The outcome of the game is then the mean weight of the edges on this cycle. The first player wants to maximise and the second player to minimise this outcome. This game is a finite perfect information two-person

*Supported in part by the ESPRIT Basic Research Action Programme of the EC under contract No. 7141 (project ALCOM II).

†Department of Computer Science, School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel. E-mail address: zwick@math.tau.ac.il.

‡Department of Computer Science, University of Warwick, Coventry CV4 7AL, England. E-mail address: Mike.Paterson@dcs.warwick.ac.uk.

game and so, by definition, has a value. Ehrenfeucht and Mycielski [EM79] show that the value ν of this finite game is also the value of the infinite game described above. Furthermore, they show, surprisingly perhaps, that both players have positional optimal strategies for the finite game. The positional optimal strategies of the finite game are also positional optimal strategies for the infinite game.

Gurvich, Karzanov and Khachiyan [GKK88], unaware of the work of Ehrenfeucht and Mycielski [EM79], considered a slightly wider class of mean payoff games which they refer to as *cyclic games*. A cyclic game is played on a directed graph $G = (V, E)$ with a weight function $w : E \to \{-W, \ldots, 0, \ldots, W\}$. The vertices of G are again divided into two classes V_1 and V_2. This need not be a bipartite partition however. The players again form a path starting at a fixed vertex $a_0 \in V$. Whenever the endpoint of the path formed is in V_1, the first player chooses the next edge; whenever it is in V_2, the second player makes the choice. Note that a player may be able, or may be forced, to play a few times in succession. The goal of the first player is again to maximise, and of the second to minimise, the long-term average weight of the path formed. Gurvich *et al.* [GKK88] note that a general theorem of Moulin [Mou76] on stationary optimal strategies in stochastic games implies that both players of a cyclic game have positional optimal strategies. The theorem of Moulin is proved non-constructively using a fixed point theorem. Gurvich *et al.* [GKK88] give an exponential time algorithm for finding such positional optimal strategies thereby giving a constructive proof of their existence. Cyclic games *with prohibitions*, a further generalisation of cyclic games, were considered by Karzanov and Lebedev [KL93]. Generalizations of mean payoff games to n players, where $n > 2$, have been considered by Alpern [Alp91].

Ehrenfeucht and Mycielski [EM79] give no efficient algorithm for finding optimal strategies for the finite and infinite games. Gurvich, Karzanov and Khachiyan [GKK88] give, as mentioned, an exponential time algorithm for these tasks. We complement their works by exhibiting an $O(|V|^3 \cdot |E| \cdot W)$ time algorithm for finding the values of the mean payoff games played on a graph $G = (V, E)$ with vertex classes V_1 and V_2. The graph G need not be bipartite so our algorithm applies to the slightly wider class of games considered by Gurvich *et al.* [GKK88]. The algorithm finds the values of all the vertices of the graph; games starting at different vertices may have different values, of course. We also give an $O(|V|^4 \cdot |E| \cdot \log(|E|/|V|) \cdot W)$ time algorithm for finding positional optimal strategies for both players. Our algorithm is polynomial in the size of the graph but only pseudo-polynomial in the weights. Our algorithm is polynomial if the weights are presented in unary notation. In particular, our algorithms work in polynomial time if the weights are taken from, say, $\{-1, 0, +1\}$. This is already a non-trivial case.

At the end of [GKK88], there is a claim that there exists a polynomial time algorithm for finding values and optimal strategies of cyclic games. According to Karzanov (personal communication), this claim was made by mistake. Lozovanu [Loz91],[Loz93] also considers cyclic games and claims a strongly polynomial time algorithm for them. He describes a simple reduction from cyclic games to simple acyclic games. Unfortunately, his reduction is not valid: the first player in his acyclic games gains some control over the length of the cycles formed in the cyclic games. His reduction fails, for example, on the complete bipartite graph with vertex sets $V_1 = \{v_0, v_2\}$, $V_2 = \{v_1, v_3\}$ and edge weights given by $w = 0$ for edges $(0,1), (1,2), (2,3), (3,0)$, $w = 1$ for $(1,0), (3,2)$, and $w = -1$ for $(0,3), (2,1)$.

We also consider situations in which one player knows in advance the positional strategy the other player is going to use. Using a result of Karp [Kar78] we show that an optimal counter-

strategy can be found in strongly polynomial time. This immediately implies that the decision problem associated with the game is in NP$\cap$co-NP. Similar observations were made by Karzanov and Lebedev [KL93].

The decision problem corresponding to mean payoff games (MPG's) is thus in NP$\cap$co-NP as well as in $\tilde{\mathrm{P}}$ (pseudo-polynomial time), but is not yet known to be in P. This gives the MPG problem a rare status shared only by a few number-theoretic problems, such as *primality* [Pra75].

Condon [Con92] has recently studied the complexity of *simple stochastic games* (SSG's) introduced originally by Shapley [Sha53]. Condon shows that the decision problem corresponding to SSG's is also in NP $\cap$ co-NP. While MPG's are deterministic, SSG's are games of chance. We describe a simple reduction from MPG's to SSG's in two steps. We first describe a reduction from MPG's to *discounted payoff games* (DPG's), and then a reduction from DPG's to SSG's.

The reduction from MPG's to SSG's shows that SSG's are at least as hard as MPG's. It also supplies an alternative proof that the MPG problem is in NP $\cap$ co-NP, though we believe that the MPG problem is strictly easier then the SSG problem. As attempts to obtain polynomial time algorithms for SSG's have not yet borne fruit, it may be interesting to focus attention on the possibly easier problem of obtaining a polynomial time algorithm for MPG's.

Various path-forming games, such as the many different versions of *geography* were studied by Bodlaender [Bod93], Fraenkel and Simonson [FS93] and Fraenkel, Scheinerman and Ullman [FSU93]. Many of these games are PSPACE-complete. It is therefore somewhat surprising that the mean payoff games that we are considering do have relatively efficient algorithms.

Mean payoff games arise naturally when trying to design algorithms for various *on-line* problems. Some possible applications of mean payoff games are described in Section 7.

The rest of the paper is organized as follows. In the next section we describe an algorithm for finding the values of a game. In Section 3 we describe an algorithm for finding optimal strategies. In Section 4 we consider the case of playing against a known positional strategy. In Section 5 we introduce discounted payoff games (DPG's) and describe a reduction from MPG's to DPG's. In Section 6 we describe the simple stochastic games (SSG's) studied by Condon [Con92] and present a reduction from DPG's to SSG's. In Section 7 we describe some applications of mean payoff games. We end in Section 8 with some concluding remarks and open problems.

2 Finding the values of a game

Let $G = (V_1, V_2, E)$ be the graph on which the game is to be played, where V_1 are the vertices of the first player and V_2 are the vertices of the second player, let $w : E \to \{-W, \ldots, 0, \ldots, W\}$ be a weight function on its edges, and $|V| = n$, where $V = V_1 \cup V_2$. Recall that the graph G need not be bipartite, there may be edges between different vertices of V_1 and between different vertices of V_2.

Our first goal is to find, for each vertex $a \in V$, the value $\nu(a)$ of the finite and infinite games that start at a. The proof, given by Ehrenfeucht and Mycielski, that the values of the finite and infinite games are equal, extends easily to the case in which the graph $G = (V_1, V_2, E)$ is not bipartite. To reach this goal we consider a third version of the game. This time the two players play the game for exactly k steps constructing a path of length k, and the weight of this path is the outcome of the game. The length of the game is known in advance to both players. We let $\nu_k(a)$ be the value of this game started at vertex $a \in V$, where player I or II plays first according to whether $a \in V_1$ or $a \in V_2$.

Theorem 2.1 *The values $\nu_k(a)$, for every $a \in V$, can be computed in $O(k \cdot |E|)$ time.*

Proof: It is easy to see that for every $a \in V$ and every $k \geq 1$ we have

$$\nu_k(a) = \begin{cases} \max_{(a,b)\in E}\{w(a,b) + \nu_{k-1}(b)\} & \text{if } a \in V_1 \ , \\ \min_{(a,b)\in E}\{w(a,b) + \nu_{k-1}(b)\} & \text{if } a \in V_2 \ . \end{cases}$$

Clearly, $\nu_0(a) = 0$ for every $a \in V$. The values $\nu_k(a)$, for every $a \in V$, can be easily computed using these recursive formulae in $O(k \cdot |E|)$ time. □

It seems intuitively clear that $\lim_{k\to\infty} \nu_k(a)/k = \nu(a)$, where $\nu(a)$ is the value of the infinite game that starts at a. The next theorem states that this is indeed the case. In the proof of this theorem we rely on the result, proved by Ehrenfeucht and Mycielski and by Gurvich, Karzanov and Khachiyan, that both players have positional optimal strategies. A *positional strategy for player I* is just a mapping $\pi_1 : V_1 \to V$ such that $(a_1, \pi_1(a_1)) \in E$ for every $a_1 \in V_1$. Similarly, a *positional strategy for player II* is a mapping $\pi_2 : V_2 \to V$ such that $(a_2, \pi_2(a_2)) \in E$ for every $a_2 \in V_2$.

Theorem 2.2 *For every $a \in V$ we have:*

$$k \cdot \nu(a) - 2nW \ \leq \ \nu_k(a) \ \leq \ k \cdot \nu(a) + 2nW \ .$$

Proof: Let $\pi_1 : V_1 \to V_2$ be a positional optimal strategy for player I in the finite game starting at a. We show that if player I plays using the strategy π_1 then the outcome of a k-step game is at least $(k-n) \cdot \nu(a) - nW$. Consider a game in which player I plays according to π_1. Push (copies of) the edges played by the players onto a stack. Whenever a cycle is formed, it follows from the fact that π_1 is an optimal strategy for player I in the finite game, that the mean weight of the cycle formed is at least $\nu(a)$. The edges that participate in that cycle lie consecutively at the top of the stack. They are all removed and the process continues. Note that at each stage the stack contains at most n edges and the weight of each of them is at least $-W$. Player I can therefore ensure that the total weight of the edges encountered in a k-step game starting from a is at least $(k-n) \cdot \nu(a) - nW$. This is at least $k \cdot \nu(a) - 2nW$ as $\nu(a) \leq W$.

Similarly, if player II plays according to a positional optimal strategy $\pi_2 : V_2 \to V_1$ of the finite game that starts at a, she can make sure that the mean of each cycle closed is at most $\nu(a)$. At most n edges are left on the stack and the weight of each of them is at most W. She can therefore ensure that the total weight of the edges encountered in a k-step game starting at a is at most $(k-n) \cdot \nu(a) + nW \leq k \cdot \nu(a) + 2nW$. □

We can now describe the algorithm for computing the exact values of the finite and infinite games.

Theorem 2.3 *Let $G = (V_1, V_2, E)$ be a directed graph and let $w : E \to \{-W, \ldots, 0, \ldots, W\}$ be a weight function on its edges. The value $\nu(a)$, for every $a \in V$, corresponding to the infinite and finite games that start at all the vertices of V can be computed in $O(|V|^3 \cdot |E| \cdot W)$ time.*

Proof: Compute the values $\nu_k(a)$, for every $a \in V$, for $k = 4n^3W$. This can be done, according to Theorem 2.1, in $O(|V|^3 \cdot |E| \cdot W)$ time. For each vertex $a \in V$, compute an estimate $\nu'(a) = \nu_k(a)/k$. By Theorem 2.2, we get that

$$\nu'(a) - \frac{1}{2n(n-1)} < \nu'(a) - \frac{2nW}{k} \leq \nu(a) \leq \nu'(a) + \frac{2nW}{k} < \nu'(a) + \frac{1}{2n(n-1)} \ .$$

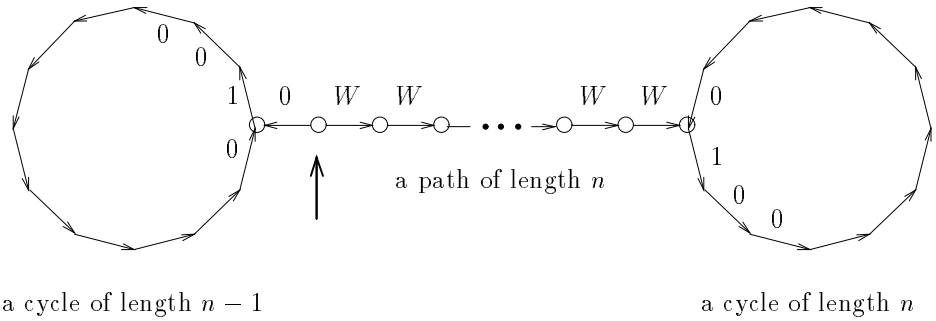

Figure 1: An example in which $k = \Omega(n^3 \cdot W)$ is needed.

The value $\nu(a)$ is a rational number with a denominator whose size is at most n. The minimum distance between two possible values of $\nu(a)$ is at least $1/n(n-1)$. The exact value of $\nu(a)$ is therefore the unique rational number with a denominator of size at most n that lies in the interval $(\nu'(a) - \frac{1}{2n(n-1)}, \nu'(a) + \frac{1}{2n(n-1)})$. This number is easily found. □

The example given in Figure 1 shows that to obtain the correct values using the algorithm described above it may be necessary to take $k = \Omega(n^3 W)$. Slightly less accuracy is needed if we just want to know whether the value of each position is negative, zero or positive. This decision problem can therefore be decided more efficiently.

Theorem 2.4 *Let $G = (V_1, V_2, E)$ be a directed graph and let $w : E \to \{-W, \ldots, 0, \ldots, W\}$ be a weight function on its edges. Let T be an integer threshold. A decision whether $\nu(a) < T$, $\nu(a) = T$, or $\nu(a) > T$, for every $a \in V$, can be made in $O(|V|^2 \cdot |E| \cdot W)$ time.*

Proof: The distance between T and the closest rational number with a denominator of size at most n is $1/n$. It is therefore enough to compute the values $\nu_k(a)$ for $k = 4n^2 W$, and this takes only $O(|V|^2 \cdot |E| \cdot W)$ time. □

3 Finding the optimal strategies

Given an algorithm for finding the value of any vertex of a graph, positional optimal strategies can be found using a simple method, which successively eliminates sets of edges using a 'group testing' technique.

Theorem 3.1 *Let $G = (V_1, V_2, E)$ be a directed graph and let $w : E \to \{-W, \ldots, 0, \ldots, W\}$ be a weight function on its edges. Positional optimal strategies for both players, for games played on this graph, can be found in $O(|V|^4 \cdot |E| \cdot \log(|E|/|V|) \cdot W)$ time.*

Proof: Start by computing the values $\nu(a)$ for every $a \in V$. If all the vertices $a \in V_1$ have outdegree one, then player I has a unique strategy and this strategy is positional and optimal. Otherwise, consider any vertex $a \in V_1$ with outdegree $d > 1$. Remove any $\lceil d/2 \rceil$ of the edges leaving a, and recompute the value of a, $\nu'(a)$ say, for the resulting graph. If $\nu'(a) = \nu(a)$ then there is a positional optimal strategy for the player I which does not use any of the removed edges; if $\nu'(a) \neq \nu(a)$ then there is a positional optimal strategy for this player using one of the removed

edges. Whichever is the case, we can now restrict attention to a subgraph G' with at least $\lfloor d/2 \rfloor$ fewer edges. Let $d(a)$ be the initial outdegree of vertex $a \in V$. After $O(\sum_{a \in V_1} \log d(a))$ such experiments we are left with a positional optimal strategy for player I. A positional optimal strategy for player II is found in a similar way. As $\sum_{a \in V} \log d(a) \leq |V| \cdot \log(|E|/|V|)$, we get that the complexity of this algorithm is $O(|V|^4 \cdot |E| \cdot \log(|E|/|V|) \cdot W)$, as required. □

An interesting open problem is whether finding positional optimal strategies is harder than just computing the values of a game. The algorithm we describe calls the full value-finding algorithm repeatedly, but uses only the value at a single vertex and ignores any information about the optimal moves of the players in the truncated games. Unfortunately, optimal moves in the truncated games may not conform to positional strategies. We think however that it should be possible to use the additional information gathered and improve our algorithm.

The running times of all the algorithms described so far depend on the size of the weights. This dependency can be avoided at the (high) price of an exponential running time in the size of the graph.

Theorem 3.2 *Let $G = (V_1, V_2, E)$ be a directed graph with a real weight function $w : E \to R$. Let $V = V_1 \cup V_2$. Positional optimal strategies for both players, for games played on this graph, can be found in $2^{O(|E|)}$ or $2^{O(|V| \log |V|)}$ time.*

Proof: It is easy to see that each player has at most $2^{O(|E|)}$ positional strategies. The values of all the vertices in V when players I and II play according to specific positional strategies, $\pi_1 : V_1 \to V$ and $\pi_2 : V_2 \to V$, are easily found in $O(|E| + |V|)$ time. We can therefore construct, in $2^{O(|E|)}$ time, a $2^{O(|E|)} \times 2^{O(|E|)}$ matrix with all the possible outcomes of the game when both players use positional strategies. Let $\nu_{\pi_1, \pi_2}(a)$ be the outcome of a game that starts at $a \in V$ in which the two players use strategies π_1 and π_2 respectively. The results of Ehrenfeucht and Mycielski and of Gurvich, Karzanov and Khachiyan imply that $\nu(a) = \max_{\pi_1} \min_{\pi_2} \{\nu_{\pi_1, \pi_2}(a)\}$, for every $a \in V$. The values of all the positions can therefore be found in $2^{O(|E|)}$ time. The results of the above mentioned authors also imply that there exists a positional strategy π_1 for the first player for which $\min_{\pi_2} \{\nu_{\pi_1, \pi_2}(a)\} = \nu(a)$, for every $a \in V$. Each such strategy is a positional optimal strategy for player I. A positional optimal strategy for player II is found similarly. The $2^{O(|V| \log |V|)}$ time algorithm is obtained in the same way since the number of positional strategies that each player may have is also bounded by $2^{O(|V| \log |V|)}$. □

4 Playing against a known positional strategy

In this section we consider degenerate games in which there is only one edge out of each vertex for player II, say. This corresponds, for example, to cases in which player I knows in advance the positional strategy according to which player II is going to play. The simple observations made in this section are very similar to observations made by Gurvich, Karzanov and Khachiyan [GKK88] and by Karzanov and Lebedev [KL93].

An $O(|V| \cdot |E|)$ algorithm of Karp [Kar78] (see also [CLR90], p. 548) for finding the maximum (or minimum) mean weight cycle of a weighted graph $G = (V, E)$ supplies, almost immediately, an efficient purely combinatorial algorithm for such special cases.

Theorem 4.1 *Let $G = (V_1, V_2, E)$ be a directed graph with a real weight function $w : E \to R$ on its edges, and assume that the outdegree of each vertex $v_2 \in V_2$ is exactly one. Then, the*

values of all the vertices and a positional optimal strategy $\pi_1 : V_1 \to V$ for player I can be found in $O(|V| \cdot |E|)$ time.

Proof: The value of vertex $a \in V$ is the maximum mean weight of a cycle reachable from a. We begin therefore by finding the strongly connected components and the component graph of G. This can be done in $O(|E| + |V|)$ time (see, e.g., [CLR90]). Next, we use Karp's algorithm to find the maximum mean weight cycle in each such strongly connected component. This takes $O(|V| \cdot |E|)$ time. We then find, again in $O(|V| \cdot |E|)$ time, the transitive closure of the component graph of G. The maximum mean weight cycle reachable from each vertex of G is then easily found in $O(|V|^2)$ time. □

The maximum (or minimum) mean weight cycle in a graph $G = (V, E)$ with relatively small weights can be found more efficiently using the scaling algorithms of Orlin and Ahuja [OA92] and Young and Tarjan and Orlin [YTO91]. If there are only two different edge weights then it can be found even faster, using an algorithm of Butkovic and Cuninghame-Green [BC92].

Could methods used by Karp's algorithm, or by the other maximum mean weight cycle algorithms, be used to obtain a more efficient algorithm for the general case? Could scaling methods be used to speed our algorithm?

The natural decision problem corresponding to MPG's is the following. Given a MPG G and a number ν, is the value of G at least ν? As a Corollary to Theorem 4.1 we get the following result.

Theorem 4.2 *The decision problem corresponding to mean payoff games is in NP ∩ co-NP.*

Proof: To show that the value of a game is at least ν, all we have to do is *guess* a positional optimal strategy for player I. We can then check, using Karp's algorithm, that the value of the game is at least ν. To show that the value of the game is less than ν, all we have to do is guess a positional optimal strategy for player II, and use Karp's algorithm to check that the value is less than ν. □

An alternative, more 'efficient', proof of Theorem 4.2 makes use of *potentials*.

Theorem 4.3 *Let $G = (V_1, V_2, E)$ be a directed graph, let $w : E \to R$ be a weight function on its edges and let $a \in V = V_1 \cup V_2$. Then, the value of the mean payoff game that starts at a is at least ν if and only if there exist subsets $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ with $a \in U = U_1 \cup U_2$, and a potential function $h : U \to R$ that satisfy the following two conditions:*

$$\begin{array}{ll} (i) & \forall u \in U_1 \;\; \exists (u,v) \in E \quad v \in U \;\wedge\; h(u) + w(u,v) \geq h(v) + \nu \;, \\ (ii) & \forall u \in U_2 \;\; \forall (u,v) \in E \quad v \in U \;\wedge\; h(u) + w(u,v) \geq h(v) + \nu \;. \end{array}$$

Proof: For the 'if' direction, suppose that such sets and potential function exist. Player I then always chooses an edge that satisfies condition (i). This makes sure that the game never leaves the set U and that the mean of every cycle formed is at least ν.

For the 'only if' direction, suppose that $\nu(a) \geq \nu$. If we subtract ν from each of the edge weights, we obtain a corresponding game where $\nu(a) \geq 0$, so we may assume $\nu(a) \geq \nu = 0$ without loss of generality. Let $\pi_1 : V_1 \to V$ be a positional optimal strategy for player I, and consider the

subgraph $G' = (U_1, U_2, E')$ of G whose vertices and edges are just those which are reachable from vertex a when player I plays according to the positional strategy π_1 and player II plays arbitrarily. We define $h(u) \in R$ for each $u \in U_1 \cup U_2$ as the minimum weight of a path from a to u, i.e., the *distance* from a to u in the weighted graph formed. Since π_1 is an optimal strategy for player I assuring that $\nu(a) \geq 0$, the weight of any cycle in G' is nonnegative and these distances are well defined. Conditions (i) and (ii) follow immediately from the definitions of G' and h. □

To show that the value of a game is at least ν, all we have to do is guess the subsets U_1 and U_2 and the potential function h. The two conditions can then be verified in linear time. A dual condition can be used to verify that the value of the game is at most ν.

5 Discounted payoff games

In this section we describe a *discounted* version of mean payoff games. This (fourth) variant, which is also interesting in its own right, will serve in the next section as a link between mean payoff games and simple stochastic games.

Let $0 < \lambda < 1$ be a real number. The weight of the i-th edge, e_i, chosen by the players is now multiplied by $(1-\lambda)\lambda^i$ and the outcome of the game is defined to be $(1-\lambda)\sum_{i=0}^{\infty} \lambda^i w(e_i)$. The goal of the first player is again to maximise the outcome of the game and the goal of the second player is to minimise this outcome. The number λ is called the *discounting factor* of the game.

Let $G = (V_1, V_2, E)$ be a directed graph and let $w : E \to R$ be a weight function on its edges. As always, we assume that the outdegree of all the vertices is at least one. Let $V = V_1 \cup V_2 = \{1, 2, \ldots, n\}$. Let $x_i = x_i(\lambda)$ be the value of a discounted game started at i. If $(i,j) \in E$, we use w_{ij} as an abbreviation for $w((i,j))$.

Theorem 5.1 *The value vector $\mathbf{x} = (x_1, \ldots, x_n)$ of the discounted games played on the graph $G = (V_1, V_2, E)$ is the unique solution of the following set of equations:*

$$x_i = \begin{cases} \max_{(i,j)\in E} \{(1-\lambda)w_{ij} + \lambda x_j\}, & \text{if } i \in V_1 \ ; \\ \min_{(i,j)\in E} \{(1-\lambda)w_{ij} + \lambda x_j\}, & \text{if } i \in V_2 \ . \end{cases}$$

Proof: Let $\mathcal{F}$ be a mapping that receives a vector $\mathbf{x}$ and returns the vector $\mathbf{y}$ such that

$$y_i = \begin{cases} \max_{(i,j)\in E} \{(1-\lambda)w_{ij} + \lambda x_j\}, & \text{if } i \in V_1 \ ; \\ \min_{(i,j)\in E} \{(1-\lambda)w_{ij} + \lambda x_j\}, & \text{if } i \in V_2 \ . \end{cases}$$

The given set of equations can be expressed in the form $\mathbf{x} = \mathcal{F}(\mathbf{x})$. If we let $\|\mathbf{v}\| = \max_i\{v_i\}$ be the *max* norm, then

$$\forall \mathbf{u}, \mathbf{v}, \ \|\mathcal{F}(\mathbf{u}) - \mathcal{F}(\mathbf{v})\| \leq \lambda \|\mathbf{u} - \mathbf{v}\| \ .$$

Thus, since $0 < \lambda < 1$, $\mathcal{F}$ is a contraction mapping with respect to the norm. It follows easily that the limit $\mathbf{x} = \lim_{n\to\infty} \mathcal{F}^n(\mathbf{0})$ exists and is the unique solution to the equation $\mathbf{x} = \mathcal{F}(\mathbf{x})$.

Let $\mathbf{x}$ be the solution of the equation $\mathbf{x} = \mathcal{F}(\mathbf{x})$. It is easy to verify that if player I plays according to a strategy which at each vertex $i \in V_1$ chooses an edge $(i,j) \in E$ which maximises $(1-\lambda)w_{ij} + \lambda x_j$, then the outcome of the game that starts from each vertex i is at least x_i. Similarly, if player II plays according to a strategy which at each vertex $i \in V_2$ chooses an edge

$(i,j) \in E$ which minimises $(1-\lambda)w_{ij} + \lambda x_j$, then the outcome of the game that starts from each vertex i is at most x_i. It follows that the value of the game starting from i is exactly x_i. $\Box$

It follows immediately from this theorem that both players of the discounted game again have positional optimal strategies. The proof in this case is much simpler than the proofs given by Ehrenfeucht and Mycielski [EM79] and by Gurvich, Karzanov and Khachiyan [GKK88] for non-discounted games.

Theorem 5.1 suggests a way of finding the values of the discounted payoff games played on a graph $G = (V_1, V_2, E)$. We are not aware, however, of any strongly polynomial time algorithm for finding a solution to the set of equations that appear in the theorem. A pseudo-polynomial time algorithm for finding the values and optimal positional strategies for discounted payoff games, similar to the algorithm presented in the proof of Theorem 2.3, can be easily devised.

Let $\nu(\lambda)$ be the value of the discounted game with discounting factor λ. As λ tends to 1, we expect $\nu(\lambda)$ to tend to ν, the value of the non-discounted game. This follows from the next theorem.

Theorem 5.2 *Let $G = (V_1, V_2, E)$ be a graph on n vertices, let $V = V_1 \cup V_2$, let $w : E \to \{-W, \ldots, 0, \ldots, W\}$ be a weight function on its edges and let λ be a real number satisfying $0 < \lambda < 1$. If $\nu(\lambda)$ and ν are the values of the discounted and mean payoff games played on the graph $G = (V_1, V_2, E)$ starting at $a \in V$, then*

$$\nu - 2n(1-\lambda)W \;\leq\; \nu(\lambda) \;\leq\; \nu + 2n(1-\lambda)W \;.$$

Proof: Consider the outcome of a discounted game in which player I uses a positional optimal strategy for the *non-discounted* game and player II uses a positional optimal strategy to counter the strategy of player I. The outcome of such a game clearly supplies a lower bound on the value $\nu(\lambda)$ of the discounted game. The play in such a case consists of a path of length k, followed by a cycle of length ℓ which is repeated indefinitely, where $0 \leq k \leq n-1$, $1 \leq \ell \leq n$ and $k + \ell \leq n$.

Assume for the moment that all the edge weights are non-negative. Let $w_0, \ldots, w_{\ell-1}$ be the weights of the edges in the cycle formed. As player I uses an optimal strategy for the non-discounted game we get that $\sum_{i=0}^{\ell-1} w_i \geq \ell\nu$. The outcome of the discounted game is then at least

$$\begin{aligned}(1-\lambda)\lambda^k(\sum_{i=0}^{\ell-1} w_i\lambda^i)(\sum_{j=0}^{\infty}\lambda^{j\ell}) &= \frac{(1-\lambda)\lambda^k}{1-\lambda^\ell}\cdot\sum_{i=0}^{\ell-1} w_i\lambda^i \\ &\geq \frac{(1-\lambda)\lambda^{k+\ell-1}}{1-\lambda^\ell}\cdot\sum_{i=0}^{\ell-1} w_i \;\geq\; \frac{\ell(1-\lambda)}{1-\lambda^\ell}\cdot\lambda^{k+\ell-1}\cdot\nu \;.\end{aligned}$$

As $\ell(1-\lambda)/(1-\lambda^\ell) > 1$ and $\lambda^{k+\ell-1} > \lambda^n > 1 - n(1-\lambda)$, this is at least $(1-n(1-\lambda))\cdot\nu$.

We now return to the general case in which the edge weights are not assumed to be non-negative. By adding W to each weight, we can make all the weights non-negative. The value and outcome of the game are changed by exactly W. Applying the previous inequality to the resulting non-negative game we get that

$$(\nu(\lambda) + W) \geq (1-n(1-\lambda))(\nu + W) \;,$$

or equivalently that

$$\nu(\lambda) \geq \nu - n(1-\lambda)(\nu + W) \geq \nu - 2n(1-\lambda)W \ .$$

The opposite inequality is proved in a similar way. □

In particular, if we choose $\lambda = 1 - 1/(4n^3W)$, then it is easy to verify that $|\nu(\lambda) - \nu| \leq 1/(2n(n-1))$, and ν can be obtained from $\nu(\lambda)$ by rounding to the nearest rational with a denominator less than n, as was done in Section 2. We thus obtain a reduction from MPG's to discounted payoff games (DPG's).

6 Reduction to simple stochastic games

In this section we describe a simple polynomial reduction from discounted payoff games (DPG's) to simple stochastic games (SSG's). This reduction, combined with the reduction from MPG's to DPG's, shows that SSG's are at least as hard as MPG's. We believe that MPG's are in fact easier than SSG's.

A *simple stochastic game* is a two-person game played on a directed graph $G = (V, E)$ whose vertex set V is the union of three disjoint sets $V_{\max}, V_{\min}$ and V_{average}. The graph also contains a special start vertex and two special vertices called the 0-sink and the 1-sink. Each edge emanating from an 'average' vertex has a rational probability attached to it. The probabilities attached to all the edges from each average vertex add up to 1.

A token is initially placed on the start vertex of the graph. At each step of the game the token is moved from a vertex to one of its neighbours, according to the following rules:

1. At a max vertex, player I chooses the edge along which the token is moved.
2. At a min vertex, player II chooses this edge.
3. At an average vertex, the edge along which the token is moved is chosen randomly according to the probabilities attached to the outgoing edges.

The game ends when the token reaches one of the sink vertices. Player I wins if the token reaches the 1-sink and player II wins otherwise, i.e., if the token reaches the 0-sink or if the game does not end. The *value* of such a game is the probability that player I wins the game when both players play optimally. As was the case for mean payoff games, the two players of a simple stochastic game have positional optimal strategies.

Simple stochastic games were first studied by Shapley [Sha53]. Many variants of them have been studied since then (see Peters and Vrieze [PV87] for a survey). Condon [Con92] was the first to study simple stochastic games from a complexity theory point of view. She showed that the natural decision problem corresponding to SSG's (i.e., given a game G and a rational number $0 < \alpha \leq 1$, is the value of G at least α?) is in NP ∩ co-NP. No polynomial time algorithm for SSG's is yet known. Some exponential algorithms for the problem are described in [MC90]. A subexponential randomized algorithm for SSG's was recently obtained by Ludwig [Lud95].

Condon [Con92] actually shows containment in NP ∩ co-NP of the decision problem that corresponds to SSG's of the following restricted form. The outdegree of each non-sink vertex is exactly two and the probability attached to each edge that emanates from an average vertex is 1/2. She then describes a reduction from general SSG's to SSG's of this restricted form. Her reduction, however, is not polynomial. A general SSG on n vertices in which the denominators of all the (rational) probabilities are at most m is transformed into a restricted SSG of size

polynomial in n and m, rather than in n and $\log m$. Her transformation can be easily modified however, as we show next, to yield a polynomial reduction.

It is easy to transform a SSG into an equivalent SSG in which the outdegree of each non-sink vertex is exactly two. Each vertex of fan-out k is simply replaced by a binary tree with k leaves. This increases the size of the graph (i.e., the number of vertices and edges) by only a constant factor. The remaining problem is therefore the simulation of binary average vertices with non-equal probabilities. Suppose we want to implement an average vertex u with two emanating edges (u, v_1) and (u, v_2), labelled respectively by the probabilities p/q and $(q-p)/q$, where p and q are integers and $2^{t-1} \leq q < 2^t$. Let $a_1 a_2 \ldots a_{t-1} a_t$ and $b_1 b_2 \ldots b_{t-1} b_t$ be the binary representations of p and $q-p$ respectively, where a_1 and b_1 are the most significant digits. We use the construct shown in Figure 2. All the vertices used are average vertices with equal probabilities. For every i, $2 \leq i \leq t+1$, there are two emanating edges that are reached from u with probability 2^{-i}. If $a_i = 1$ then connect one of the edges with probability $2^{-(i+1)}$ to v_1, and if $b_i = 1$ then connect one of these edges to v_2. All the unused edges are connected back to u. Is it easy to check that v_1 and v_2 are eventually reached with the appropriate probabilities. The number of vertices used in this construction is proportional to the number of bits needed to represent the transition probabilities. The reduction is therefore polynomial.

A simple stochastic game is said to *halt with probability 1* if, no matter how the players play, the game ends with probability 1. The proof of the following theorem can be found in Condon [Con92]. Note the similarity of this theorem to Theorem 5.1.

Theorem 6.1 *Let $G = (V, E)$ be a SSG that halts with probability 1, and let $p(u, v)$ denote the probability attached to an edge (u, v) that emanates from an average vertex u. The values $\nu(v)$ of the vertices of G form the unique solution to the following set of equations:*

$$\nu(u) = \begin{cases} \max_{(u,v)\in E}\{\nu(v)\} & \text{if } u \text{ is a max vertex,} \\ \min_{(u,v)\in E}\{\nu(v)\} & \text{if } u \text{ is a min vertex,} \\ \sum_{(u,v)\in E}\{p(u,v)\cdot\nu(v)\} & \text{if } u \text{ is an average vertex,} \end{cases}$$

along with the conditions that $\nu(0\text{-sink}) = 0$ and $\nu(1\text{-sink}) = 1$.

We are finally in a position to describe a reduction from discounted payoff games (DPG's) to simple stochastic games (SSG's). Recall that we have already described a reduction from MPG's to DPG's.

Let $G = (V_1, V_2, E)$ be a DPG with discounting factor λ. If we add a constant c to all the weights of the game, the value of the game is increased by c. If we multiply all the weights of the game by a constant $c > 0$, the value of the game is multiplied by c. We can therefore scale the weights so that they will all be rational numbers in the interval $[0, 1]$. If the original weights were in the range $\{-W, \ldots, 0, \ldots, W\}$, then the new weights will be rational numbers with denominators and numerators in the range $\{0, 1, \ldots, 2W\}$.

We construct in the following way a SSG $G' = (V', E')$, with the same value as the scaled DPG $G = (V_1, V_2, E)$ with discounting factor λ. Each edge (u, v) with weight w in G is replaced by the construct shown in Figure 3. We let $V' = V_{\max} \cup V_{\min} \cup V_{\text{average}}$, where $V_{\max} = V_1$, $V_{\min} = V_2$ and V_{average} is the set of intermediate vertices added. The simple stochastic game G' halts with probability 1, as in each transition there is a probability of $1 - \lambda$ of reaching a sink vertex. The values of the vertices of the discounted payoff game G satisfy the set of

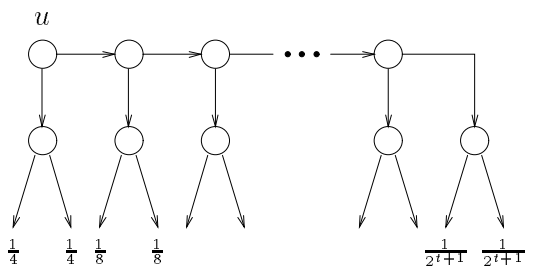

Figure 2: Implementing an average vertex with arbitrary probabilities.

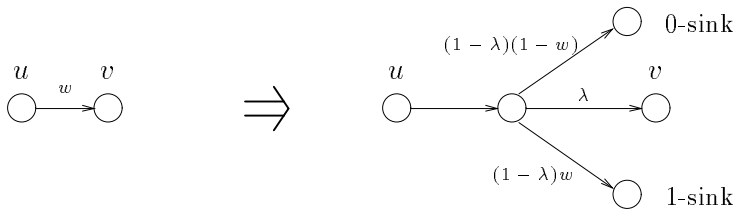

Figure 3: Simulating a transition of a discounted payoff game.

equations given in Theorem 5.1. The values of the vertices of the simple stochastic game G' satisfy the set of equations given in Theorem 6.1. These two sets of equations become *identical* once the intermediate variables, that correspond to the intermediate vertices introduced by the transformation described in Figure 3, are eliminated. As this set of equations has a unique solution, the values of the two games are equal. The transformation of G to G' can clearly be carried out in polynomial time. This completes the description of the reduction.

7 Some applications

In this section we briefly mention some applications of mean payoff games.

Consider a system with n possible states. At each time unit, the system receives one of k possible requests. The system is allowed to change its state and then it has to serve the request. The transition from state i to state j costs a_{ij}, and serving a request of type t from state i costs b_{it}. What, in the worst-case, is the average cost of serving a request?

Borodin, Linial and Saks [BLS92] performed a *competitive analysis* of such systems, which they call *on-line metrical task systems*. If we look at the worst-case instead, we get a bipartite mean payoff game $G = (V_1, V_2, E)$ played between the system and an adversary that chooses the requests. The adversary plays from the vertices of $V_1 = \{1, 2, \ldots, n\}$. The algorithm plays from the vertices of $V_2 = \{(i, j) \mid 1 \leq i \leq n \ , \ 1 \leq j \leq k\}$. The edge $i \rightarrow (i, j)$ corresponds to the request of task j while the system is in the state i. The edge $(i, j) \rightarrow k$ corresponds to the transition of the system to state k before serving this request. The weight of all the edges of the form $i \rightarrow (i, j)$ is 0. The weight of an edge $(i, j) \rightarrow k$ is $a_{ik} + b_{kj}$.

Consider finite-window on-line string matching algorithms (see [CHPZ95] for a definition).

What, in the worst-case, is the average number of comparisons that an optimal algorithm has to perform per text character? The problem can be formulated as a bipartite mean payoff game played between the designer of a string matching algorithm and an adversary that answers the queries made by an algorithm. The reward (the complement of cost) obtained by the algorithm at each stage is the amount by which it can shift its window. For each pattern string and window size we obtain a mean payoff game, the solution of which yields an optimal string matching algorithm for that pattern and window size.

As a last example, consider the problem of selection with limited storage. Suppose that we are to receive a long stream of numbers. We are supposed to select the k-th largest of these numbers. We have however only s storage locations, for some $s > k$, each one of them capable of holding a single number. Each input number must be read into one of these s storage locations before it can be compared to any of the numbers held in the other $s-1$ locations. The previous value of the cell into which the input number is read is lost. What is the average number of comparisons needed per input element in the worst case? Selection and sorting problems with limited storage were considered by Munro and Paterson [MP80]. They allowed several passes over the input stream however.

The problem can again be formulated as a bipartite mean payoff game $G = (V_1, V_2, E)$ played by the designer of a selection algorithm and an adversary that answers queries made by an algorithm. The vertices of V_1 correspond to those partial orders of s elements in which no element is known to be smaller than k elements. The vertices of V_2 coresond to such partial orders together with requests for comparing two specific elements in each such partial order. Edges from V_1 to V_2 correspond to comparison requests made by the algorithm. Edges from V_2 to V_1 correspond to the answers of the adversary. Each vertex of V_2 has two edges emanating from it, corresponding to the two possible outcomes of the comparison requested. The weight of all the edges from V_1 to V_2 is 0. The weight of each edge $u \to v$ from V_2 to V_1 is the number of elements that are known, as a result of the last comparison, to be smaller than at least k elements. Such elements are discarded and are replaced by new input elements. Twice the value of the game is the average number of elements that can be discarded as a result of a single comparison. The graph that corresponds to the selection of the second largest element using four storage locations, i.e., $k = 2$ and $s = 4$, is given in Figure 4. It is not difficult to verify that the value of this game is $\nu = 1/3$, the starting point is this case does not matter. This means that the average number of comparisons needed per input element is $3/2$.

8 Concluding remarks

Mean payoff games form a very natural class of full information games and we think that resolving their complexity is an interesting issue. We conjecture that they lie in P but, since none of the standard methods seems to yield a polynomial time algorithm for them, the study of mean payoff games may require new algorithmic techniques. If such positive approaches are unsuccessful, the example of mean payoff games may help in exploring the structure of NP $\cap$ co-NP.

Acknowledgments

We would like to thank Sergiu Hart, Ehud Lehrer, Nimrod Megiddo, Moni Naor, Noam Nisan and Avi Wigderson for helpful discussions and suggestions, Alexander Karzanov for pointing out a flaw in the previous version of Theorem 3.1, and Thomas McCormick for bringing references [GKK88] and [KL93] to our attention.

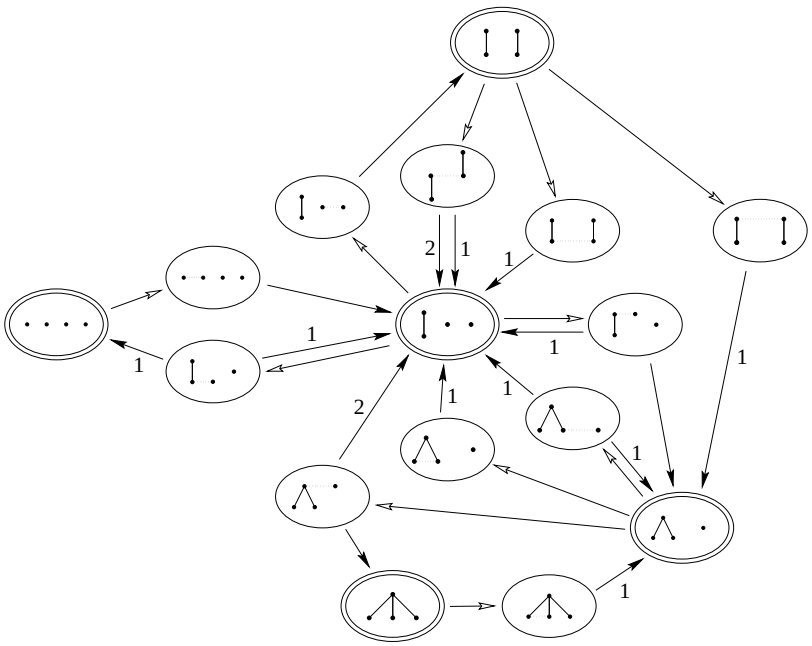

Figure 4: A mean payoff game that corresponds to the problem of selecting the second largest element with only four storage locations.

References

[Alp91] S. Alpern. Cycles in extensive form perfect information games. *Journal of Mathematical Analysis and Applications*, 159:1–17, 1991.

[BC92] P. Butkovic and R.A. Cuninghame-Green. An $O(n^2)$ algorithm for the maximum cycle mean of an $n \times n$ bivalent matrix. *Discrete Applied Mathematics*, 35:157–162, 1992.

[BLS92] A. Borodin, N. Linial, and M.E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, 1992.

[Bod93] H.L. Bodlaender. Complexity of path-forming games. *Theoretical Computer Science*, 110:215–245, 1993.

[CHPZ95] R. Cole, R. Hariharan, M. Paterson, and U. Zwick. Tighter lower bounds on the exact complexity of string matching. *SIAM Journal on Computing*, 24:30–45, 1995.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms*. The MIT Press, 1990.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, February 1992.

[EM79] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[FS93] A.S. Fraenkel and S. Simonson. Geography. *Theoretical Computer Science*, 110:197–213, 1993.

[FSU93] A.S. Fraenkel, E.R. Scheinerman, and D. Ullman. Undirected edge geography. *Theoretical Computer Science*, 112:371–381, 1993.

[GKK88] V.A. Gurvich, A.V. Karzanov, and L.G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics*, 28:85–91, 1988.

[Kar78] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[KL93] A.V. Karzanov and V.N. Lebedev. Cyclical games with prohibitions. *Mathematical Programming*, 60:277–293, 1993.

[Loz91] D.D. Lozovanu. Algorithms to solve some classes of network minimax problems and their applications. *Cybernetics*, 29:93–100, 1991.

[Loz93] D.D. Lozovanu. Strongly polynomial algorithms for finding minimax paths in networks and solution of cyclic games. *Cybernetics and Systems Analysis*, 29:754–759, 1993.

[Lud95] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.

[MC90] M. Melekopoglou and A. Condon. On the complexity of the policy iteration algorithm for stochastic games. Technical Report TR 941, University of Wisconsin, June 1990.

[Mou76] H. Moulin. Prolongement des jeux a deux joueurs de somme nulle. *Bull. Soc. Math. France, Memoires*, 45, 1976.

[MP80] J.I. Munro and M.S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[OA92] J.B. Orlin and R.K. Ahuja. New scaling algorithms for the assignment and minimum mean cycle problems. *Mathematical Programming*, 54:41–56, 1992.

[Pra75] V.R. Pratt. Every prime has a succinct certificate. *SIAM Journal on Computing*, 4(3):214–220, 1975.

[PV87] H.J.M. Peters and O.J. Vrieze. Surveys in game theory and related topics. CWI Tract 39, Centrum voor Wiskunde en Informatica, Amsterdam, 1987.

[Sha53] L.S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. U.S.A.*, 39:1095–1100, 1953.

[YTO91] N.E. Young, R.E. Tarjan, J.B. Orlin. Faster parametric shortest path and minimum-balance algorithms. *Networks*, 21:205–221, 1991.