Fast Algorithms for Energy Games in Special Cases[*]

Sebastian Forster
Department of Computer Science
University of Salzburg, Austria
sebastian.forster@plus.ac.at

Antonis Skarlatos
Department of Computer Science
University of Salzburg, Austria
antonis.skarlatos@plus.ac.at

Tijn de Vos
Department of Computer Science
University of Salzburg, Austria
tijn.devos@plus.ac.at

In this paper, we study algorithms for special cases of energy games, a class of turn-based games on graphs that show up in the quantitative analysis of reactive systems. In an energy game, the vertices of a weighted directed graph belong either to Alice or to Bob. A token is moved to a next vertex by the player controlling its current location, and its energy is changed by the weight of the edge. Given a fixed starting vertex and initial energy, Alice wins the game if the energy of the token remains nonnegative at every moment. If the energy goes below zero at some point, then Bob wins. The problem of determining the winner in an energy game lies in $\mathsf{NP} \cap \mathsf{coNP}$. It is a long standing open problem whether a polynomial time algorithm for this problem exists.

We devise new algorithms for three special cases of the problem. The first two results focus on the single-player version, where either Alice or Bob controls the whole game graph. We develop an $\tilde{O}(n^{\omega} W^{\omega})$ time algorithm for a game graph controlled by Alice, by providing a reduction to the All-Pairs Nonnegative Prefix Paths problem (APNP), where W is the maximum absolute value of any edge weight and ω is the best exponent for matrix multiplication. Thus we study the APNP problem separately, for which we develop an $\tilde{O}(n^{\omega} W^{\omega})$ time algorithm. For both problems, we improve over the state of the art of $\tilde{O}(mn)$ for small W. For the APNP problem, we also provide a conditional lower bound which states that there is no $O(n^{3-\varepsilon})$ time algorithm for any $\varepsilon > 0$, unless the APSP Hypothesis fails. For a game graph controlled by Bob, we obtain a near-linear time algorithm. Regarding our third result, we present a variant of the value iteration algorithm, and we prove that it gives an $O(mn)$ time algorithm for game graphs without negative cycles, which improves a previous upper bound. The all-Bob algorithm is randomized, all other algorithms are deterministic.

1 Introduction

Energy games belong to a class of turn-based games on graphs that show up in the quantitative analysis of reactive systems. A game graph can possibly represent a scheduling problem, where vertices are the configurations of the system and edges carry positive or negative values representing the evolution of resources. Thus, in this model resources can be consumed or produced. The energy games problem has been introduced in the early 2000s [13, 6], but also have been implicitly studied before due to its ties to mean-payoff games [21]. Energy games have applications in, among others, computer aided verification and automata theory [13, 5, 12], and in online and streaming problems [31]. From its computational perspective, the problem of determining the winner in an energy game lies in $\mathsf{NP} \cap \mathsf{coNP}$. It is an intriguing open problem whether a polynomial time algorithm for this problem exists.

An energy game is played by two players, say Alice and Bob, on a *game graph*, which is a weighted directed graph such that each vertex is either controlled by Alice or Bob. The game starts by placing a token with an initial energy on a starting vertex. The game is played in rounds, and every time the token

[*]This work is supported by the Austrian Science Fund (FWF): P 32863-N. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 947702). A full version is available at `https://arxiv.org/abs/2307.08442`.

is located at a vertex controlled by Alice, then Alice chooses the next location of the token among the outgoing edges, otherwise Bob chooses the next move. The token has an *energy level* (in the beginning this is equal to the initial energy) and every time it traverses an edge, the weight of the edge is added to the energy level (a negative weight amounts to a reduction of the energy level). The objectives of the players are as follows: Alice wants to minimize the initial energy that is necessary to keep the energy level nonnegative at all times, whereas Bob wants to maximize this value (and possibly drive it to ∞). The computational problem is to determine for each vertex the minimum initial energy such that Alice can guarantee against all choices of Bob that the energy level always stays nonnegative.

Energy games are a generalization of parity games [24, 9], polynomial-time equivalent to mean-payoff games [6, 9], and a special case of simple stochastic games [31]. Recent progress on parity games yielded several quasipolynomial time algorithms [11], but the corresponding techniques seem to not carry over to energy and mean-payoff games [20]. Consequently, the complexity of energy games is still "stuck" at pseudopolynomial [9] or subexponential time [4]. Hence, in this paper we focus on interesting special cases (which are non-trivial problems) that admit fast algorithms. Two of these cases are game graphs where all vertices are controlled by one player, and the third case are game graphs with no negative cycles.

All-Pairs Nonnegative Prefix Paths. We also study another reachability problem with energy constraints [22, 17], the *All-Pairs Nonnegative Prefix Paths (APNP) problem*. In this problem, the goal is to find for every pair of vertices whether there exists a path π such that the weight of each prefix of π is nonnegative. We use this problem to obtain the result for the special case where Alice controls the whole game graph, since the two problems are closely related. Dorfman, Kaplan, Tarjan, and Zwick [17] solve the more general problem, where for each pair of vertices the goal is to find the path π of maximum weight among all options. This problem naturally generalizes APSP, and they solve it in $O(mn + n^2 \log n)$ time.

Energy Games. The state-of-the-art algorithms for the energy games are either deterministic with running time $O\left(\min(mnW, mn2^{n/2} \log W)\right)$ [9, 18] or randomized with subexponential running time $2^{O(\sqrt{n \log n})}$ [4]. Special cases of the energy games have been studied by Chatterjee, Henzinger, Krinninger, and Nanongkai [15]. They present a variant of the value iteration algorithm of [9] with running time $O(m|A|)$, where A is a sorted list containing all possible minimum energy values. This does not improve the general case, as A in the worst case is the list $\{0, 1, \dots, nW, \infty\}$. However, it does give a faster running time if the weights adhere to certain restrictions. Moreover, they develop a scaling algorithm with running time $O(mn \log W (\log \frac{W}{P} + 1) + mn\frac{W}{P})$, where $P \in \{\frac{1}{n}, \dots, W\}$ is a lower bound on the *penalty* of the game.

For the special case where there are no negative cycles in the game graph, the penalty can be set to W, and the scaling algorithms of [15] solves the problem in $O(mn \log W)$ time. For another special case where the whole game graph is controlled by Alice, Brim and Chaloupka [8] provided an $\tilde{O}(mn)$[1] running time algorithm as a subroutine for the two-players version.

Mean-Payoff Games. In a mean-payoff game, the objective of Alice is to maximize the average of the weights of edges traversed so far, whereas Bob's objective is to minimize this mean payoff. It is well known that any energy games instance can be solved by solving $O(n \log(nW))$ mean-payoff games [6],

[1]We write $\tilde{O}(f)$ for $O(f \operatorname{poly} \log f)$.

and any mean-payoff game instance can be solved by solving $O(\log(nW))$ energy games with maximal weight nW [9][2]. Thus, any of the aforementioned algorithms for solving energy games also yields an algorithm for solving mean-payoff games at the expense of at most an additional factor of $O(n\log(nW))$ in the running time. Zwick and Paterson [31] provided the first pseudopolynomial time algorithm that computes all the mean-payoff values, with $O(mn^3W)$ running time. Later, the running time was improved by Brim, Chaloupka, Doyen, Gentiline, and Raskin [9] to $O(mn^2W\log(nW))$, using their reduction to energy games. The state-of-the-art algorithm for solving a mean-payoff game is due to Comin and Rizzi [16] which runs in $O(mn^2W)$ time.

1.1 Our Results and Techniques

All-Pairs Nonnegative Prefix Paths. The version of All-Pairs Nonnegative Prefix Paths (APNP) problem where we want to find the path of maximum weight [17], naturally generalizes the All-Pairs Shortest Paths (APSP) problem. The APSP Hypothesis states that there is no $O(n^{3-\varepsilon})$ time algorithm for the APSP, for any $\varepsilon > 0$. However, this version of APNP is more than what is necessary for the application of energy games. We show that the weaker version which only computes reachability (as APNP has been defined), also does not allow for a $O(n^{3-\varepsilon})$ time algorithm for any $\varepsilon > 0$, under the APSP Hypothesis.

Theorem 1.1. *Unless the APSP Hypothesis fails, there is no $O(n^{3-\varepsilon})$ time algorithm that solves the All-Pairs Nonnegative Prefix Paths problem, for any $\varepsilon > 0$.*

We parameterize the maximum absolute value of any edge weight W, and we obtain an algorithm with a faster running time for small values of W.

Theorem 1.2. *There exists a deterministic algorithm that, given a graph $G = (V, E, w)$ with edge weights in the interval $[-W, W]$, solves the All-Pairs Nonnegative Prefix Paths problem in $\tilde{O}(n^{\omega}W^{\omega})$ time.*

All-Alice. Our first contribution regarding the special cases of energy games, concerns the *all-Alice case* in which all vertices are controlled by Alice. Note that if we fix a strategy for Bob in any game graph, this can be seen as an all-Alice instance.

Theorem 1.3. *There exists a deterministic algorithm that, given a game graph $G = (V, E, w)$ in which all vertices are controlled by Alice, computes the minimum sufficient energy of all vertices in $\tilde{O}(n^{\omega}W^{\omega})$ time.*

Note that the aforementioned reduction from energy games to mean-payoff games always introduces Bob vertices. Thus, algorithms for the all-Alice mean-payoff decision problem cannot be leveraged by this reduction to compute the minimum energies in the all-Alice case.

Our approach for the all-Alice case consists of two steps. In the first step, we identify all vertices Z such that minimum initial energy 0 suffices, by using Theorem 1.2. In the second step, we compute the paths of least energy reaching any vertex in Z. For small values of W, this improves on the state-of-the-art $\tilde{O}(mn)$ algorithm [8].

All-Bob. Our second contribution regarding the special cases of energy games, is a faster algorithm for the *all-Bob case* in which all vertices are controlled by Bob. Note that if we fix a strategy for Alice in any game graph, this can be seen as an all-Bob instance.

[2]Unless stated otherwise, we always consider the versions of the games where we compute the mean-payoff value/minimum initial energy for *all* vertices.

Theorem 1.4. *There exists a randomized (Las Vegas) algorithm that, given a game graph $G = (V, E, w)$ in which all vertices are controlled by Bob, computes the minimum sufficient energy of all vertices, and with high probability the algorithm takes $O(m \log^2 n \log nW \log\log n)$ time.*

To the best of our knowledge, the fastest known algorithm for the all-Bob case is implied by the reduction to the mean-payoff decision problem and has a running time of $\tilde{O}(mn \log^2 W)$. This comes from $\tilde{O}(n \log W)$ calls to the state-of-the-art negative-cycle detection algorithm [3, 10].

Our approach for the all-Bob case consists of two steps. In the first step, we run a negative-cycle detection algorithm to remove all vertices reaching a negative cycle. In the second step, we add an artificial sink to the graph with an edge from every vertex to the sink, and we compute the shortest path of every vertex to the sink using a single-source shortest paths (SSSP) algorithm. Note that this construction is very close to Johnson's method for computing suitable vertex potentials [23]. Further note that, since energy games are not symmetric for Alice and Bob, our near-linear time all-Bob algorithm has no implications for the all-Alice case.

No Negative Cycles. Finally, we give an improved algorithm for the special case where there are no negative cycles.

Theorem 1.5. *There exists a deterministic algorithm that, given a grame graph $G = (V, E)$ without negative cycles, computes the minimum sufficient energy of all vertices in $O(mn)$ time.*

To the best of our knowledge, the fastest known algorithm for this special case has a running time of $O(mn \log W)$ by running the above mentioned algorithm of Chatterjee, Henzinger, Krinninger, and Nanongkai [15] with penalty $P = W$. We use a new variant of the value iteration algorithm where the energy function after i steps corresponds to the minimum energy function in an *i-round game*. A similar variant has been used by Chatterjee, Doyen, Randour, and Raskin [14] for the Mean-Payoff games. We adapt this algorithm and provide the necessary analysis to use it for energy games.

An i-round game is a finite version of the energy game, where a token is passed for only i rounds. In this version, the goal is to find the initial energy that Alice needs, in order to keep the energy level nonnegative for these i-rounds. Then we show that in graphs without negative cycle, the infinite game is equivalent to the n-round game.

Structure of the paper. In the next section, we provide some preliminaries, including the formal definition of an energy game. In Section 3, we study the All-Pairs Nonnegative Prefix Paths problem, and we present an algorithm for the special case that the edge weights are in $\{-1, 0, +1\}$, an algorithm for general edge weights, and a lower bound. Next in Section 4, we consider the all-Alice case by reducing this problem to the All-Pairs Nonnegative Prefix Paths problem. In Section 5, we consider the all-Bob case, and finally in Section 6, we consider game graphs without negative cycles.

2 Preliminaries

Graphs. Given a directed graph $G = (V, E, w)$, we denote by $n = |V|$ the number of vertices, by $m = |E|$ the number of edges, and by W the maximum absolute value of any edge weight. Also, we denote $N^+(v)$ for the *out-neighborhood* of v, i.e., $N^+(v) := \{u \in V : (v, u) \in E\}$. Further, we denote $\deg^+(v)$ for the *out-degree* of v, i.e., $\deg^+(v) := |N^+(v)|$. Similarly, $N^-(v)$ and $\deg^-(v)$ denote the *in-neighborhood* and *in-degree* respectively.

A *path* P is a sequence of vertices $u_0u_1\cdots$ such that $(u_i,u_{i+1})\in E$ for every $i\geq 0$. We say a path is *finite* if it contains a finite number of vertices (counted with multiplicity). We say a path is *simple* if each vertex appears at most once. A *lasso* is a path of the form $u_0u_1\cdots u_ju_i$, where the vertices $u_0,\dots,u_j$ are disjoint and $i<j$. In other words, it is a simple path leading to a cycle. A *nonnegative prefix path* is a path $P=u_0u_1\cdots$ such that $\sum_{j=0}^{i-1}w(u_j,u_{j+1})\geq 0$ for all $1\leq i\leq |P|$. Further, we denote the weight of a path $P=u_0u_1\cdots$ by $w(P):=\sum_{j=0}^{|P|-1}w(u_j,u_{j+1})$. For a fixed path $P=u_0u_1\cdots$, the energy level $e(u_i)$ of a vertex u_i in P is equal to $\sum_{j=0}^{i-1}w(u_j,u_{j+1})$. That is, the sum of all the weights along P until u_i.

Let $G=(V,E,w)$ be a directed graph with edge weights -1 and $+1$, and let $s,t\in V$ be two vertices of G. Then, a *Dyck path from s to t* is a nonnegative prefix path from s to t of total weight zero [7]. For a graph H, we refer to the corresponding functions by using H as subscript (e.g., we use the notation $w_H(\cdot)$ for the weight function of H).

Energy Games. An energy game is an infinite duration game played by two players, Alice and Bob. The game is played on a *game graph* which a weighted directed graph $G=(V,E,w)$, where each vertex has at least one outgoing edge. The weights are integers and lie in the range $\{-W,-W+1,\dots,W-1,W\}$. The set of vertices is partitioned in two sets V_A and V_B, controlled by Alice and Bob respectively. Furthermore, we are given a starting vertex $s\in V$, and initial energy $e_0\geq 0$. We start with position $v_0=s$. After the i_{th} round, we are at a position $v_i\in V$ and have energy e_i. In the i_{th} round, if $v_{i-1}\in V_A$ ($v_{i-1}\in V_B$) then Alice (Bob) chooses a next vertex $v_i\in N^+(v_{i-1})$ and the energy changes to $e_i=e_{i-1}+w(v_{i-1},v_i)$. The game ends when $e_i<0$, in which case we say that Bob wins. If the game never ends, namely, $e_i\geq 0$ for all $i\geq 0$, we say that Alice wins. The goal is to find out the minimum initial energy $e_0\geq 0$ such that Alice wins when both players play optimally. Note that allowing $e_0=\infty$ means that such an energy always exist.

To make this goal more formal, we have to introduce *strategies*. A strategy for Alice (Bob) tells us given the current point $v_i\in V_A$ ($v_i\in V_B$) and the history of the game, $v_0,\dots,v_i$, where to move next. It turns out that we can restrict ourselves to *positional strategies* [19, 6], which are deterministic and do not depend on the history of the game. We denote a positional strategy of Alice by $\sigma\colon V_A\to V$ where $\sigma(v)\in N^+(v)$ for $v\in V_A$, and a positional strategy of Bob by $\tau\colon V_B\to V$ where $\tau(v)\in N^+(v)$ for $v\in V_B$. For any pair of strategies (σ,τ) we define $G(\sigma,\tau)$ to be the subgraph (V,E') corresponding to these strategies, where $E'=\{(v,\sigma(v)):v\in V_A\}\cup\{(v,\tau(v)):v\in V_B\}$. Note that in this graph each vertex has exactly one out-neighbor. Let P_i be the unique path $s=u_0,u_1,\dots,u_i$ in $G(\sigma,\tau)$ of length i originating at s. Then at vertex s with initial energy e_0 and with these strategies, Alice wins if $e_0+w(P_i)\geq 0$ for all $i\geq 0$, and Bob wins if $e_0+w(P_i)<0$ for at least one $i\geq 0$. The *minimum sufficient energy at s with respect to σ and τ* is the minimum energy such that Alice wins, namely $e_{G(\sigma,\tau)}(s):=\max\{0,-\inf_{i\geq 0}w(P_i)\}$. Finally, we define the *minimum sufficient energy* at s as follows:

$$e^*_G(s):=\min_{\sigma}\max_{\tau}e_{G(\sigma,\tau)}(s),$$

where the minimization and the maximization are over all the positional strategies σ of Alice and τ of Bob, respectively. We omit the subscript G, and use $e_{\sigma,\tau}(s)$ instead of $e_{G(\sigma,\tau)}(s)$, whenever this is clear from the context. By Martin's determinacy theorem [26], we have that $\min_\sigma\max_\tau e_{\sigma,\tau}(s)=\max_\tau\min_\sigma e_{\sigma,\tau}(s)$, thus the outcome is independent of the order in which the players pick their strategy. Now we can define *optimal strategies* as follows. A strategy σ^* is an optimal strategy for Alice, if $e_{\sigma^*,\tau}(s)\leq e^*(s)$ for any strategy τ of Bob. Similarly, τ^* is an optimal strategy for Bob, if $e_{\sigma,\tau^*}(s)\geq e^*(s)$ for any strategy σ of Alice. An *energy function* is a function $e\colon V\to\mathbb{Z}_{\geq 0}\cup\{\infty\}$. The function $e^*_G(\cdot)$ (or $e^*(\cdot)$) as defined above, is the *minimum sufficient energy function*.

3 All-Pairs Nonnegative Prefix Paths Problem

In this section, we study the All-Pairs Nonnegative Prefix Paths (APNP) problem. The goal of this problem is to find for every pair of vertices whether there exists a nonnegative prefix path between them. A similar problem is the *All-Pairs Dyck-Reachability problem*, where the goal is to find for every pair of vertices whether there exists a Dyck path between them (given that the edge weights are in $\{-1,+1\}$). Furthermore, another standard problem is the *transitive closure problem*, which asks to find for every pair of vertices whether there exists a path between them.

Bradford [7] provided an $\tilde{O}(n^{\omega})$ time algorithm for the All-Pairs Dyck-Reachability problem. Moreover, the transitive closure problem admits an $\tilde{O}(n^{\omega})$ algorithm [1].

Theorem 3.1. *There exists a deterministic algorithm that, given a graph $G=(V,E,w)$ with edge weights in $\{-1,1\}$, solves the All-Pairs Dyck-Reachability problem in $\tilde{O}(n^{\omega})$ time.*

Our approach for the APNP problem consists of two stages. At first, we solve the APNP problem for the special case where the edge weights are from the set $\{-1,0,+1\}$, by exploiting the algorithm of [7] for the All-Pairs Dyck-Reachability problem. Afterwards, we extend our algorithm to work with general weights, by showing that a reduction used in [2] preserves the properties we need.

In the end of the section, we also present a conditional lower bound for the APNP problem under the APSP Hypothesis, which is one of the main hypotheses in fine-grained complexity.

3.1 All-Pairs Nonnegative Prefix Paths with edge weights in $\{-1,0,+1\}$

Consider a graph $G=(V,E)$ with edge weights -1 and $+1$. By definition, we have that any Dyck path is also a nonnegative prefix path. However, the opposite is not necessarily true. Recall that nonnegative prefix paths allow the energy level of their last vertex to be a strictly positive value, while in Dyck paths this value must be zero. This implies that an All-Pairs Dyck-Reachability algorithm does not trivially gives us an All-Pairs Nonnegative Prefix Paths algorithm. Nevertheless, we show how to overcome this issue and we use an All-Pairs Dyck-Reachability algorithm as a subroutine in order to solve the All-Pairs Nonnegative Prefix Paths problem.

Algorithm for the $\{-1,0,+1\}$ case. Consider a directed graph $G=(V,E,w)$, with edge weights in $\{-1,0,+1\}$. In the beginning of the algorithm, we construct a graph G_2 as follows.

1. Initially, we create a new graph $G_1=(V_1,E_1,w)$ by replacing every edge of zero weight with an edge of weight $+1$ and an edge of weight -1. Specifically, for each vertex u with at least one outgoing edge $(u,v)\in E$ with $w(u,v)=0$, we add a new vertex u', and add an edge (u,u') with $w(u,u')=+1$. Next, for each edge $(u,v)\in E$ with $w(u,v)=0$, we remove the edge (u,v), and add the edge (u',v) with weight -1.[3]
2. Next, we run on G_1 the algorithm of Theorem 3.1, which solves the All-Pairs Dyck-Reachability in time $\tilde{O}(n^{\omega})$ for edge weights in $\{-1,+1\}$.
3. Finally, we create another new graph $G_2=(V,E_2)$ with the original vertex set and an edge set E_2 defined as follows. The set E_2 contains an edge $(u,v)\in V\times V$ if and only if there is a Dyck path from u to v in G_1 or $w(u,v)=1$ in G, if $(u,v)\in E$.

[3]Note that by doing the naive thing which is to replace each edge of zero weight by two edges, one with weight $+1$ and one with weight -1, potentially blows up the number of vertices to $\Omega(m)$. In turn, since the running time depends on the number of vertices, this translates to a blow up of the running time.

In the end, we run on G_2 a transitive closure algorithm, and we return that there is a nonnegative prefix path in G if and only if there is a path in G_2. Notice that graphs G and G_1 are weighted, while G_2 is unweighted.

Analysis of the algorithm. The following observation shows that the replacement of zero weight edges is valid, in the sense that nonnegative prefix paths of total weight zero[4] in G correspond to Dyck paths in G_1 and vice versa. Moreover, we prove the claim that the transitive closure problem in G_2 is equivalent to the All-Pairs Nonnegative Prefix Paths problem in G.

Observation 3.2. *For every pair of vertices $u, v \in V$, there exists a nonnegative prefix path of total weight zero from u to v in G if and only if there exists a Dyck path from u to v in G_1.*

Lemma 3.3. *For every pair of vertices $u, v \in V$, there exists a nonnegative prefix path from u to v in G if and only if there exists a path from u to v in G_2.*

Proof. Assume that there exists a nonnegative prefix path π from u to v in G. Let a be the first vertex after u along π with a minimum energy level. Initially, we show that the edge (u,a) appears in G_2. Since π is a nonnegative prefix path, we have that $e(u) \leq e(a)$. If $e(u) < e(a)$, then there must be an edge (u,a) in G with weight $+1$. Also if $e(u) = e(a)$, then the subpath of π from u to a is a nonnegative prefix path of total weight zero. Then by Observation 3.2, the subpath of π from u to a is a Dyck path in G_1. Therefore, in both cases we have added the edge (u,a) in G_2. As the vertex a has a minimum energy level, we can apply the same argument iteratively starting from a, to conclude that there exists a path from u to v in G_2.

Assume now that there exists a path π from u to v in G_2. By construction, the edges of π correspond either to edges in G with weight $+1$ or to Dyck paths in G_1. By Observation 3.2, these Dyck paths in G_1 correspond to nonnegative prefix paths of total weight zero in G. Since positive edges increase the energy level and nonnegative prefix paths at least maintain the energy level, we conclude that there exists a nonnegative prefix path from u to v in G. ☐

Lemma 3.4. *There exists a deterministic algorithm that, given a graph $G = (V, E, w)$ with edge weights in $\{-1, 0, +1\}$, solves the All-Pairs Nonnegative Prefix Paths problem in $\tilde{O}(n^{\omega})$ time.*

Proof. The number of vertices of G_1 is $O(n)$ by construction, where n is the initial number of vertices in G. Hence, the construction of G_2 runs in $\tilde{O}(n^{\omega})$. Moreover, the transitive closure problem in G_2 can be solved in $\tilde{O}(n^{\omega})$ time as well [1]. Thus by Lemma 3.3, the claim follows. ☐

3.2 All-Pairs Nonnegative Prefix Paths with general edge weights

We extend now Lemma 3.4 for graphs with general edge weights, in the cost of an extra factor W^{ω} in the running time. The idea is to use the reduction by Alon, Galil, and Margalit [2], who reduce the All-Pairs Shortest Paths (APSP) problem with general edge weights to the special case where the edge weights are in $\{-1, 0, +1\}$. We present the reduction for completeness, and we prove that the same reduction also preserves the properties that we need for the All-Pairs Nonnegative Prefix Paths problem.

[4]Observe that a Dyck path is a nonnegative prefix path of total weight zero consisting only of edges -1 and $+1$. Thus, we avoid to use the term *Dyck path* for G because it may contains edges of weight zero.

Reduction from general weights to $\{-1,0,+1\}$ [2]. Given a graph $G=(V,E,w)$ with weights in the interval $[-W,W]$, we create another graph G' with weights only in $\{-1,0,+1\}$, as follows. For every vertex $v \in V$ in G, we create $2W+1$ vertices $\{v^i\}_{i=-W}^{W}$ in G'. We say that vertex v^0 of G' is the origin of vertex v. Then, we add in G' an edge (v^{i+1},v^i) of weight -1, for every $-W \leq i \leq -1$, and an edge (v^{i-1},v^i) of weight 1, for every $1 \leq i \leq W$. Moreover, for every edge (u,v) of weight k in G, we add an edge (u^k,v^0) of zero weight in G'.

Theorem 1.2. *There exists a deterministic algorithm that, given a graph $G=(V,E,w)$ with edge weights in the interval $[-W,W]$, solves the All-Pairs Nonnegative Prefix Paths problem in $\tilde{O}(n^{\omega}W^{\omega})$ time.*

Proof. The idea is to apply the reduction mentioned above and use the algorithm of Lemma 3.4 in G'. Then, we claim that there exists a nonnegative prefix path from u to v in G if and only if there exists a nonnegative prefix path from u^0 to v^0 in G'.

Regarding the running time, since the number of vertices of the new graph G' after the reduction becomes $\Theta(nW)$, the running time of the algorithm becomes $\tilde{O}((nW)^{\omega})$. It remains to prove the correctness of the algorithm.

Let π be a nonnegative prefix path from u to v in G. We construct a path π' from u^0 to v^0 in G' as follows. For every edge $(a,b) \in \pi$ of weight k, we add to π' the unique subpath from a^0 to b^0 of weight k in G', which exists by construction. Since π is a nonnegative prefix path in G, and every subpath we add to π' consists either only of edges with weight in $\{-1,0\}$ or only of edges with weight in $\{0,+1\}$, we can infer that π' is a nonnegative prefix path from u^0 to v^0 in G'.

For the other direction, let π' be a nonnegative prefix path from u^0 to v^0 in G'. We construct a path π from u to v in G as follows. Let a^0 be the first vertex after u^0 along π' such that, a^0 is the origin vertex of a different vertex than u (i.e., a^0 is the origin of a vertex $a \neq u$). By construction, there exists an edge (u,a) of weight k in G, where k is the weight of the subpath from u^0 to a^0 in π'. We add the edge (u,a) in π, and continue with the construction of π by applying the same argument iteratively starting from a^0 until we reach v^0. Since π' is a nonnegative prefix path in G', and each prefix of π corresponds to a prefix in π', we can infer that π is a nonnegative prefix path from u to v in G.

Therefore, the pair of vertices $\{u^0,v^0\}$ in G' contains the information for the pair of vertices $\{u,v\}$ in G, and so the claim follows. □

3.3 Lower bound for All-Pairs Nonnegative Prefix Paths

We prove a lower bound on the running time of All-Pairs Nonnegative Prefix Paths problem under the APSP Hypothesis. The APSP Hypothesis is an assertions that the All-Pairs Shortest Paths (APSP) problem cannot be solved in truly subcubic $O(n^{3-\varepsilon})$ time, for any $\varepsilon > 0$. Vassilevska Williams and Williams [29] proved that APSP and Negative Triangle are equivalent under subcubic reductions. The Negative Triangle problem is defined as follows. Given a graph $G=(V,E,w)$, the goal is to find three vertices a,b,c such that $w(a,b)+w(b,c)+w(c,a)<0$, that is, the vertices a,b,c form a negative weight cycle.

Recently, a reduction from the Negative Triangle problem to the h-hop-bounded s-t path problem was given by Polak and Kociumaka [25], in order to prove a hardness result for the latter. Motivated by this reduction, we also reduce the Negative Triangle problem to the All-Pairs Nonnegative Prefix Paths problem to obtain a hardness result for the All-Pairs Nonnegative Prefix Paths problem, as shown in Theorem 1.1.

We first provide an auxiliary lemma, which we also use later in Lemma 4.1.

Lemma 3.5. *Given a graph $G = (V,E,w)$, let C be a nonnegative weight cycle in G (i.e., $w(C) \geq 0$). Then, there is a vertex $u \in C$ in the cycle, such that there exists a nonnegative prefix path in G from u to itself along C.*

Proof. Let $Q \subseteq C$ be a subpath of C with the most negative total weight, and Q' be the rest of C (i.e., $Q \cup Q' = C$). Notice that the weight of all prefixes in Q' must be nonnegative, otherwise this negative weight prefix could be merged with Q, contradicting the fact that Q is the subpath of C with the most negative total weight. Moreover, as $w(C) \geq 0$ we have that $w(Q') \geq -w(Q)$. Since by definition of Q, there is no prefix of Q with more negative total weight, it holds that $Q' \cup Q$ is a nonnegative prefix path from the first vertex of Q' to itself along C. □

Theorem 1.1. *Unless the APSP Hypothesis fails, there is no $O(n^{3-\varepsilon})$ time algorithm that solves the All-Pairs Nonnegative Prefix Paths problem, for any $\varepsilon > 0$.*

Proof. Consider a Negative Triangle instance $G = (V,E)$. We create a directed graph $G_1 = (V_1, E_1)$ as follows. The vertex set V_1 of G_1 consists of five copies of all vertices, i.e., $V_1 := \{v^i : v \in V, i \in \{1,2,3,4,5\}\}$. For every edge $(u,v) \in E$ of weight $w(u,v)$, we add an edge (u^i, v^{i+1}) to E_1 with weight $-w(u,v)$, for $1 \leq i < 4$. Also for each vertex $v \in V$, we add an edge (v^4, v^5) of weight $w_{\min} = -1$.

We claim that there exists a negative weight triangle in G if and only if there is a vertex $v \in V$ such that there exists a nonnegative prefix path from v^1 to v^5 in G_1. In this case, since the reduction is subcubic and the time to check all vertices in G_1 is $O(n)$, an $O(n^{3-\varepsilon})$ time algorithm for the All-Pairs Nonnegative Prefix Paths problem would imply an $O(n^{3-\varepsilon})$ time algorithm for the Negative Triangle problem, for any $\varepsilon > 0$, contradicting the APSP Hypothesis.

We proceed with the proof of the claim. Suppose that there are three vertices a,b,c that form a negative weight cycle C in G, and let G_2 be the graph G after flipping the sign of the weights. Then we have that $w_{G_2}(C) > 0$ in G_2, and based on Lemma 3.5 there is a vertex $v \in C$, such that there exists a nonnegative prefix path in G_2 from v to itself along C. Notice that v can be either a,b or c, and by construction, the paths $a^1b^2c^3a^4a^5$, $b^1c^2a^3b^4b^5$, and $c^1a^2b^3c^4c^5$ exist in G_1. Thus without loss of generality, we can assume that v is a and we use the path $a^1b^2c^3a^4a^5$ in G_1. By construction, it holds that $w_{G_1}(a^1,b^2) = w_{G_2}(a,b), w_{G_1}(b^2,c^3) = w_{G_2}(b,c)$, $w_{G_1}(c^3,a^4) = w_{G_2}(c,a)$ and $w_{G_1}(a^4,a^5) = w_{\min}$. The path $abca$ is a nonnegative prefix path in G_2, and so the path $a^1b^2c^3a^4$ is a nonnegative prefix path in G_1 as well. Moreover since $w_{G_2}(C) > 0$, we have that $w_{G_2}(C) \geq -w_{\min}$, which implies that:

$$w_{G_1}(a^1,b^2) + w_{G_1}(b^2,c^3) + w_{G_1}(c^3,a^4) \geq -w_{\min}.$$

Thus, we can conclude that the path $a^1b^2c^3a^4a^5$ is a nonnegative prefix path in G_1.

For the other direction, let $a^1b^2c^3a^4a^5$ be a nonnegative prefix path in G_1. By construction of G_1 and the fact that G does not contain self-loops, it must be the case that the corresponding vertices a,b,c must be pairwise different in G. By definition of a nonnegative prefix path, it holds that:

$$w_{G_1}(a^1,b^2) + w_{G_1}(b^2,c^3) + w_{G_1}(c^3,a^4) \geq -w_{\min} > 0.$$

By construction, we have that $w(a,b) = -w_{G_1}(a^1,b^2), w(b,c) = -w_{G_1}(b^2,c^3)$ and $w(c,a) = -w_{G_1}(c^3,a^4)$. Therefore, it is true that $w(a,b) + w(b,c) + w(c,a) < 0$, and the vertices a,b,c form a negative weight cycle in G. □

4 The All-Alice Case

In this section, we develop an algorithm that computes the minimum sufficient energies of all vertices for game graphs controlled by Alice. In particular, we obtain the following result.

Theorem 1.3. *There exists a deterministic algorithm that, given a game graph $G = (V, E, w)$ in which all vertices are controlled by Alice, computes the minimum sufficient energy of all vertices in $\tilde{O}(n^{\omega} W^{\omega})$ time.*

The idea is to use the algorithm of Theorem 1.2 for the All-Pairs Nonnegative Prefix Paths problem. Hélouët, Markey, and Raha [22] provide a relevant reduction from the problem of whether zero energy suffices to the problem of whether there exists a nonnegative prefix path. Hence, one idea would be to apply this reduction and run the algorithm of Theorem 1.2. Unfortunately this reduction affects the weights, and the maximum weight of the new instance can be as big as mW, which in turn affects the running time of the algorithm.

To that end, we present another way to use the All-Pairs Nonnegative Prefix Paths algorithm of Theorem 1.2 without affecting the maximum weight of the graph. The algorithm consists of two phases. In the first phase, we detect all the vertices such that initial zero energy suffices, and in the second phase we compute the minimum sufficient energy for the rest of the vertices.

In the first phase of the algorithm, initially we run the All-Pairs Nonnegative Prefix Paths algorithm of Theorem 1.2 on the game graph $G = (V, E, w)$. Hence, we retrieve the information of whether there exists a nonnegative prefix path from a vertex u to a vertex v, for any two vertices $u, v \in V \times V$. Then for each vertex $v \in V$, we check whether there is a vertex u (including v) such that there exists a nonnegative prefix path from v to u and from u to u. If this is the case, then we add this vertex to a set Z. The next lemma shows that the set Z is actually the set of all vertices such that initial energy zero suffices.

Lemma 4.1. *The set Z is the same as the set $\{v \in V : e^*(v) = 0\}$, and is computed in $\tilde{O}(n^{\omega} W^{\omega})$ time.*

Proof. Suppose that the algorithm adds a vertex v to Z. Then, there must be a vertex u (possibly $u = v$) such that there exists a nonnegative prefix path from v to u and from u to u. By merging then these two paths, and by definition of minimum sufficient energy, we can conclude that $e^*(v) = 0$.

Suppose now that the minimum sufficient energy of a vertex $v \in V$ is zero (i.e., $e^*(v) = 0$). By definition of minimum sufficient energy, there must exist a nonnegative prefix lasso P which contains a nonnegative cycle C. Also by Lemma 3.5, there is a vertex $u \in C$ in the cycle, such that there exists a nonnegative prefix path from u to itself. As a result, the algorithm finds these vertices v and u and adds v to Z.

The running time of this process is dominated by the running time of the All-Pairs Nonnegative Prefix Paths algorithm, which is $\tilde{O}(n^{\omega} W^{\omega})$ based on Theorem 1.2. □

The set Z can be seen as the set of possible vertices to 'end' in. Any optimal strategy would still have to define how to move from such a vertex $v \in Z$, but since we know that $e^*(v) = 0$, there has to be a path such that from this vertex no initial energy is necessary. So the goal of the second phase, is to find for each vertex $v \in V \setminus Z$ the best way to hit a vertex in Z. The following lemma shows that this comes down to a shortest path computation. Brim and Chaloupka [8] use a similar idea inside their subroutine for the Mean-Payoff games.

Lemma 4.2. *Given a game graph $G = (V, E, w)$ where all vertices belong to Alice and the set $Z := \{v \in V : e^*(v) = 0\}$ is known, we can compute the remaining minimum sufficient energies through a single SSSP computation in G.*

For the proof we refer to the full version of the paper. Together, Lemma 4.1 and Lemma 4.2 prove Theorem 1.3, by using also the fact that we can compute SSSP deterministically in $\tilde{O}(n^{\omega}W)$ time [27, 30].

5 The All-Bob Case

In this section, we restrict ourselves to the case where all vertices belong to Bob. We show that this special case admits a near-linear time algorithm, by essentially reducing the problem to detecting negative cycles and computing shortest paths. We obtain the following result.

Theorem 1.4. *There exists a randomized (Las Vegas) algorithm that, given a game graph $G = (V,E,w)$ in which all vertices are controlled by Bob, computes the minimum sufficient energy of all vertices, and with high probability the algorithm takes $O(m\log^2 n\log nW\log\log n)$ time.*

Proof. We split the algorithm and proof in two parts, depending on who wins the game in a particular vertex. The first part of the algorithm consists of identifying the vertices with infinite energy (namely, the vertices where Bob wins), and the second part consists of calculating the finite energies of the remaining vertices (namely, the vertices where Bob loses).

Vertices where Bob wins. First, we identify the vertices where Bob wins, i.e., the vertices v with $e^*(v) = \infty$. Hereto, we decompose G in to strongly connected components $C_1,\dots,C_r$, for some $r \geq 1$. On each C_i, we run a negative cycle detection algorithm. If there is a negative cycle, we set $e(v) = \infty$ for all $v \in C_i$. Next we find the vertices that can reach these cycles. Let $A := \{v \in V : e(v) = \infty\}$ be the union of the strongly connected components with a negative cycle. Then from A we run an inward reachability algorithm (e.g., DFS, BFS) towards each vertex v and if there is a path from v to A, we set $e(v) = \infty$. In the correctness proof, we show that $e(v) = \infty$ if and only if Bob wins at v.

Correctness. For any vertex $v \in V$, Bob wins if and only if there is a path from v to a negative cycle. Let v be a vertex where Bob wins, and let $C^{(v)}$ be the negative cycle reachable from v. If v belongs to the strongly connected component of $C^{(v)}$, then our algorithm outputs $e(v) = \infty$. If v belongs to a different connected component, then the path to the negative cycle is detected in the inward reachability algorithm and we also output $e(v) = \infty$.

Suppose we output $e(v) = \infty$. If we do this because v belongs to a strongly connected in which we detected a negative cycle, then clearly there is path from v to the negative cycle, and hence Bob wins at v. If we set $e(v) = \infty$ because there is a path from v to A, then there is a path from v towards a strongly connected component containing a negative cycle, and hence to a negative cycle itself. So again Bob wins at v.

Running time. We can decompose G in to strongly connected components in $O(m)$ time [28]. On each connected component C_i, we can detect whether there is a negative cycle in the graph in $O(|E(C_i)|\log^2 n\log nW\log\log n)$ time w.h.p. [10], thus the total time is $O(m\log^2 n\log nW\log\log n)$ w.h.p. The inward reachability algorithm can be implemented by a simple DFS or BFS in $O(m)$ time. Hence in total we obtain w.h.p a running time of $O(m\log^2 n\log nW\log\log n)$ for this part.

Vertices where Bob loses. Second, we compute the correct value for the vertices where Bob loses, i.e., the vertices v with $e(v) < \infty$. Note that for this part we can restrict ourselves to the subgraph where we omit all vertices with $e(v) = \infty$. We also add a new sink vertex t to the graph, and for every $v \in V$ we insert an edge (v,t) with $w(v,t) = 0$. Now for each vertex v, we compute the minimum distance $d(v,t)$ from v to t, and we set $e(v) = \max\{-d(v,t), 0\}$. In the correctness proof, we show that $e^*(v) = e(v)$ for each $v \in V$ with $e(v) < \infty$.

Correctness. Consider now a vertex v such that $e(v) < \infty$. First we show that $e^*(v) \geq e(v)$. Let u be the last vertex (excluding t itself) on the shortest path from v to t, and $P_{v,u}$ be the corresponding prefix from v to u. Then Bob can choose to move along the path $P_{v,u}$ forcing Alice to use at least $\max\{-w(P_{v,u}),0\}$ initial energy. As $d(v,t) = w(P_{v,u}) + w(u,t) = w(P_{v,u}) + 0 = w(P_{v,u})$, we conclude that Alice needs at least $\max\{-d(v,t),0\} = e(v)$ initial energy.

It remains to show $e^*(v) \leq e(v)$. Since there are no negative cycles, by definition we have that $e^*(v) = \max\{-\min_{u \in V} w(P_u),0\}$, where the minimization is over all the simple paths from v to u. Also for all $u \in V$, it holds that $d(v,u) \leq w(P_u)$ and $d(v,t) \leq d(v,u) + w(u,t) = d(v,u) + 0 = d(v,u)$. Thus we get that $e^*(v) = \max\{-\min_{u \in V} w(P_u),0\} \leq \max\{-\min_{u \in V} d(v,t),0\} = \max\{-d(v,t),0\} = e(v)$.

Running time. To compute the shortest paths from v to t, we flip the direction of all the edges and we compute the minimum distances from t to v in the new graph. This clearly corresponds to the minimum distances from v to t in the original graph. Since this computation is the negative weight single source shortest path problem, it can be done in $O(m \log^2 n \log nW \log\log n)$ time w.h.p. [10]. $\square$

6 Game Graphs Without Negative Cycles

In this section, we provide an $O(mn)$ time algorithm for the special case where the game graph has no negative cycles. We do this in three steps: first, we introduce a finite duration energy game, where a token is passed for i rounds. The goal is to compute for each vertex, the minimum initial energy that Alice needs in order to keep the energy nonnegative for those i rounds. Second, we provide an algorithm that computes this value in $O(mi)$ time. Finally, we show that for graphs with no negative cycles, it suffices to find this minimum initial energy for a game of n rounds.

6.1 Finite Duration Games

We introduce a version of the energy game that lasts i rounds. We define strategies and energy functions analogous to the infinite duration game, as in Section 2. A strategy for Alice is a function $\sigma_i : V^*V_A \to V$, such that for all finite paths $u_0u_1 \cdots u_j$ with $j < i$ and $u_j \in V_A$, we have that $\sigma_i(u_0u_1 \cdots u_j) = v$ for some edge $(u_j, v) \in E$. Similarly we define a strategy τ_i for Bob by replacing V_A with V_B. A path $u_0u_1 \cdots u_j$ of length j is consistent with respect to strategies σ_i and τ_i, if $\sigma_i(u_0u_1 \cdots u_k) = u_{k+1}$ for all $u_k \in V_A$ and $\tau_i(u_0u_1 \cdots u_k) = u_{k+1}$ for all $u_k \in V_B$, where $0 \leq k < j \leq i$. The minimum sufficient energy at a vertex u corresponding to strategies σ_i and τ_i is defined as $e_{\sigma_i,\tau_i}(u) := \max\{-\min w(P),0\}$, where the minimization is over all the consistent paths P with respect to σ_i and τ_i of length at most i originating at u. The minimum sufficient energy at a vertex u is defined as follows:

$$e_i^*(u) := \min_{\sigma_i} \max_{\tau_i} e_{\sigma_i,\tau_i}(u),$$

where we minimize over all strategies σ_i for Alice and maximize over all strategies τ_i for Bob. As for the infinite duration game, we know by Martin's determinacy theorem [26] that $\min_{\sigma_i} \max_{\tau_i} e_{\sigma_i,\tau_i}(u) = \max_{\tau_i} \min_{\sigma_i} e_{\sigma_i,\tau_i}(u)$. Now we define *optimal strategies* as follows. A strategy σ_i^* is an optimal strategy for Alice at a vertex u, if for any strategy τ_i for Bob it holds that $e_{\sigma_i^*,\tau_i}(u) \leq e_i^*(u)$. Likewise a strategy τ_i^* is an optimal strategy for Bob at a vertex u, if for any strategy σ_i for Alice it holds that $e_{\sigma_i,\tau_i^*}(u) \geq e_i^*(u)$. A value $e(u)$ is a sufficient energy at a vertex u, if there exists a strategy σ_i such that for any strategy τ_i, it holds that $e_{\sigma_i,\tau_i}(u) \leq e(u)$. In this case, observe that the following is true:

$$e_i^*(u) = \max_{\tau_i} e_{\sigma_i^*,\tau_i}(u) \leq \max_{\tau_i} e_{\sigma_i,\tau_i}(u) \leq e(u).$$

Next, we show the following lemma about the minimum energy function, a similar version has also been used for the infinite duration game in [9] and [15]. For the proof, see the full version of the paper.

Lemma 6.1. *Given a game of i rounds and a vertex $u \in V$, the energy $e_i^*(u)$ satisfies the following properties:*

$$\text{if } u \in V_A \text{ then } \exists v \in N^+(u) : e_i^*(u) + w(u,v) \geq e_{i-1}^*(v) \tag{1}$$

$$\text{if } u \in V_B \text{ then } \forall v \in N^+(u) : e_i^*(u) + w(u,v) \geq e_{i-1}^*(v) \tag{2}$$

6.2 A Value Iteration Algorithm for Finite Duration Games

In this section, we present Algorithm 1, a value iteration algorithm for a game lasting i rounds that computes for each vertex $u \in V$ the value $e_i^*(u)$. We note that Algorithm 1 consists of i steps, where at every step each edge is scanned at most once. Clearly this means the algorithm takes $O(mi)$ time.

Algorithm 1 Value iteration algorithm for an i-round game

Input: A game graph $G = (V, E, w, \langle V_A, V_B \rangle)$, a number of iterations i

Output: The minimum sufficient energy $e_i(u)$ of each $u \in V$, in order to play the game for i rounds

```
1  $\forall u \in V : e_0(u) \leftarrow 0$
2  for $j = 1$ to $i$ do
3      foreach $u \in V$ do
4          if $u \in V_A$ then
5              $e_j(u) \leftarrow \max\{\min_{(u,v)\in E}\{e_{j-1}(v) - w(u,v)\}, 0\}$
6          end
7          if $u \in V_B$ then
8              $e_j(u) \leftarrow \max\{\max_{(u,v)\in E}\{e_{j-1}(v) - w(u,v)\}, 0\}$
9          end
10     end
11 end
12 return $e_i$
```

Lemma 6.2. *Let $e_i(\cdot)$ be the function returned by Algorithm 1, then $e_i(u) = e_i^*(u)$ for all $u \in V$.*

Proof. We prove the claim by induction on i, which is both the number of steps of the algorithm and the duration of the game.

Base case: For $i = 0$ steps, the algorithm sets for each $u \in V : e_0(u) = 0 = e_0^*(u)$.

Inductive Step: We assume that after $i-1$ steps $e_{i-1}(u) = e_{i-1}^*(u)$, and we prove that after i steps $e_i(u) = e_i^*(u)$ as well. We first show that $e_i(u) \geq e_i^*(u)$.

Consider the case that $u \in V_A$. Let v' be the neighbor that minimizes the relation in the i_{th} step in Line 5. Then it holds that $e_i(u) + w(u,v') \geq e_{i-1}(v')$. Using the edge (u,v') with initial energy $e_i(u)$, Alice can move to v' with remaining energy at least $e_{i-1}(v')$. By the inductive hypothesis it holds that $e_{i-1}(v') = e_{i-1}^*(v')$, so there exists an optimal strategy σ_{i-1}^* such that for any strategy τ_{i-1}, we have that $e_{\sigma_{i-1}^*,\tau_{i-1}}(v') \leq e_{i-1}(v')$. Define the strategy σ_i in the following way: $\forall x \in V^*V_A : \sigma_i(ux) = \sigma_{i-1}^*(x)$ and $\sigma_i(u) = v'$. Then we get a strategy σ_i such that for any strategy τ_i, it holds that $e_{\sigma_i,\tau_i}(u) \leq e_i(u)$. This implies that $e_i(u)$ is a sufficient energy at vertex u, and so $e_i(u) \geq e_i^*(u)$.

Consider the case that $u \in V_B$. Due to the i_{th} step in Line 8, it holds that $e_i(u) + w(u,v) \geq e_{i-1}(v)$, for all $v \in N^+(u)$. Hence for any choice of a neighboring edge (u,v) with initial energy $e_i(u)$, Bob moves to a neighbor v with remaining energy at least $e_{i-1}(v)$. By the inductive hypothesis, for all $v \in N^+(u)$ it holds that $e_{i-1}(v) = e^*_{i-1}(v)$, so there exists an optimal strategy σ^*_{i-1} such that for any strategy τ_{i-1}, we have that $e_{\sigma^*_{i-1},\tau_{i-1}}(v) \leq e_{i-1}(v)$. Define the strategy σ_i in the following way: $\forall x \in V^*V_A : \sigma_i(ux) = \sigma^*_{i-1}(x)$. Then we get a strategy σ_i such that for any strategy τ_i, it holds that $e_{\sigma_i,\tau_i}(u) \leq e_i(u)$. This implies that $e_i(u)$ is a sufficient energy at vertex u, and so $e_i(u) \geq e^*_i(u)$.

It remains to show that $e_i(u) \leq e^*_i(u)$. Consider the case that $u \in V_A$. If $e_i(u) = 0$ then the claim holds trivially. If $e_i(u) > 0$, then based on Line 5, we have that $e_i(u) + w(u,v) \leq e_{i-1}(v)$ for all $v \in N^+(u)$. By Lemma 6.1, there exists $v' \in N^+(u)$ such that $e^*_i(u) + w(u,v') \geq e^*_{i-1}(v')$, which means that:

$$e_i(u) + w(u,v') \leq e_{i-1}(v') = e^*_{i-1}(v') \leq e^*_i(u) + w(u,v') \;\Rightarrow\; e_i(u) \leq e^*_i(u),$$

where the equality holds by the inductive hypothesis.

Consider the case that $u \in V_B$. If $e_i(u) = 0$ then the claim holds trivially. Otherwise based on Line 8, there exists $v' \in N^+(u)$ such that $e_i(u) + w(u,v') = e_{i-1}(v')$. By Lemma 6.1, we have that $e^*_i(u) + w(u,v) \geq e^*_{i-1}(v)$ for all $v \in N^+(u)$, which means that:

$$e_i(u) + w(u,v') = e_{i-1}(v') = e^*_{i-1}(v') \leq e^*_i(u) + w(u,v') \;\Rightarrow\; e_i(u) \leq e^*_i(u),$$

where the equality holds by the inductive hypothesis. □

6.3 No Negative Cycles

The goal of this section is to show that for graphs with no negative cycles, it holds that $e^*_n(u) = e^*(u)$, for all $u \in V$. Hereto, we show in Lemma 6.4 that as in the infinite duration game, positional strategies suffice when no negative cycles are present. In the proof, we use the following alternative characterization of $e_{\sigma_i,\tau_i}(u)$.

Let σ_i and τ_i be strategies for Alice and Bob respectively, and let $u \in V$ be a vertex. Moreover, let $u_0u_1\cdots u_j$ be the consistent path of length j with respect to σ_i and τ_i, where $u_0 = u$. Then given an initial energy e_{init}, the energy level at vertex u_j is equal to the value $e_{\text{init}} + \sum_{k=0}^{j-1} w(u_k, u_{k+1})$. We denote $e^*_{\text{init}}(u)$ for the minimum nonnegative initial energy such that the energy level at each vertex of the corresponding consistent path of length i, is nonnegative. The following lemma shows that $e_{\sigma_i,\tau_i}(u) = e^*_{\text{init}}(u)$ (for the proof, see the full version of the paper).

Lemma 6.3. *For a vertex u and two fixed strategies σ_i and τ_i, let P be the consistent path with respect to σ_i and τ_i of length i originating at u. Then it holds that $e_{\sigma_i,\tau_i}(u) = e^*_{\text{init}}(u)$.*

Now we are ready to show that positional strategies suffice in graphs without negative cycles. For the proof see the full version of the paper.

Lemma 6.4. *Consider a graph with no negative cycles and a game of i rounds. Then for the minimum sufficient energy $e^*_i(u)$ at a vertex $u \in V$, it suffices for both players to play positional strategies.*

We use this fact to show that a game of n rounds is equivalent to a game of infinite duration for a game graph without negative cycles.

Lemma 6.5. *Consider a graph with no negative cycles. Then for each vertex $u \in V$, the minimum sufficient energy needed at u for a game of n rounds, is equal to the minimum sufficient energy needed at u for a game of infinite rounds. In other words, $e^*_n(u) = e^*_\infty(u) = e^*(u)$ for all $u \in V$.*

Proof. Let σ and τ be two arbitrary positional strategies for the infinite duration game. By definition, we have that $e_{\sigma,\tau}(u) = \max\{-\min w(P), 0\}$, where the minimization is over all the consistent paths with respect to σ and τ originating at u. Since the graph contains only nonnegative cycles and the strategies are positional, the path that minimizes the relation is a simple path, and so, its length is at most n. Hence it follows that $e_{\sigma,\tau}(u) = \max\{-\min_{|P|\leq n} w(P), 0\}$. In turn, this is equivalent to using positional strategies for a game of n rounds. Hence it holds that $e_{\sigma,\tau}(u) = e_{\sigma_n,\tau_n}(u)$, where σ_n and τ_n are the strategies σ and τ respectively, restricted to the first n rounds. This implies that $e^*(u) = \min_{\sigma_n} \max_{\tau_n} e_{\sigma_n,\tau_n}(u)$, where σ_n and τ_n are positional strategies for a game of n rounds. By Lemma 6.4, this equals $e^*_n(u)$ and the claim follows. ☐

Together, Lemma 6.2 and Lemma 6.5 prove Theorem 1.5.

References

[1] Alfred V. Aho, John E. Hopcroft & Jeffrey D. Ullman (1974): *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

[2] Noga Alon, Zvi Galil & Oded Margalit (1997): *On the Exponent of the All Pairs Shortest Path Problem*. J. Comput. Syst. Sci. 54(2), pp. 255–262, doi:10.1006/jcss.1997.1388.

[3] Aaron Bernstein, Danupon Nanongkai & Christian Wulff-Nilsen (2022): *Negative-Weight Single-Source Shortest Paths in Near-linear Time*. In: 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022, IEEE, pp. 600–611, doi:10.1109/FOCS54457.2022.00063.

[4] Henrik Björklund & Sergei G. Vorobyov (2007): *A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games*. Discrete Applied Mathematics 155(2), pp. 210–229, doi:10.1016/j.dam.2006.04.029. Announced at MFCS 2004.

[5] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger & Barbara Jobstmann (2009): *Better Quality in Synthesis through Quantitative Objectives*. In: Proc. of the 21st International Conference on Computer Aided Verification (CAV 2009), Lecture Notes in Computer Science 5643, Springer, pp. 140–156, doi:10.1007/978-3-642-02658-4_14. arXiv:0904.2638.

[6] Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey & Jirí Srba (2008): *Infinite Runs in Weighted Timed Automata with Energy Constraints*. In Franck Cassez & Claude Jard, editors: Proc. of the 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2008), Lecture Notes in Computer Science 5215, Springer, pp. 33–47, doi:10.1007/978-3-540-85778-5_4.

[7] Phillip G. Bradford (2017): *Efficient exact paths for dyck and semi-dyck labeled path reachability (extended abstract)*. In: Proc. of the 8th IEEE Annual Conference on Ubiquitous Computing, Electronics and Mobile Communication (UEMCON 2017), IEEE, pp. 247–253, doi:10.1109/UEMCON.2017.8249039.

[8] Luboš Brim & Jakub Chaloupka (2012): *Using strategy improvement to stay alive*. International Journal of Foundations of Computer Science 23(03), pp. 585–608, doi:10.1142/S0129054112400291.

[9] Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini & Jean-François Raskin (2011): *Faster algorithms for mean-payoff games*. Formal Methods in System Design 38(2), pp. 97–118, doi:10.1007/s10703-010-0105-x. Announced at MEMICS 2009 and GAMES 2009.

[10] Karl Bringmann, Alejandro Cassis & Nick Fischer (2023): *Negative-Weight Single-Source Shortest Paths in Near-Linear Time: Now Faster!* arXiv preprint arXiv:2304.05279, doi:10.48550/arXiv.2304.05279.

[11] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li & Frank Stephan (2022): *Deciding Parity Games in Quasi-polynomial Time*. SIAM Journal on Computing 51(2), pp. 17–152, doi:10.1137/17m1145288. Announced at STOC 2017.

[12] Pavol Cerný, Krishnendu Chatterjee, Thomas A. Henzinger, Arjun Radhakrishna & Rohit Singh (2011): *Quantitative Synthesis for Concurrent Programs*. In: *Proc. of the 23rd International Conference on Computer Aided Verification (CAV 2011)*, *Lecture Notes in Computer Science* 6806, Springer, pp. 243–259, doi:10.1007/978-3-642-22110-1_20. arXiv:1104.4306.

[13] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger & Mariëlle Stoelinga (2003): *Resource Interfaces*. In Rajeev Alur & Insup Lee, editors: *Proc. of the Third International Conference on Embedded Software (EMSOFT 2003)*, *Lecture Notes in Computer Science* 2855, Springer, pp. 117–133, doi:10.1007/978-3-540-45212-6_9.

[14] Krishnendu Chatterjee, Laurent Doyen, Mickael Randour & Jean-François Raskin (2015): *Looking at mean-payoff and total-payoff through windows*. *Inf. Comput.* 242, pp. 25–52, doi:10.1016/j.ic.2015.03.010.

[15] Krishnendu Chatterjee, Monika Henzinger, Sebastian Krinninger & Danupon Nanongkai (2014): *Polynomial-Time Algorithms for Energy Games with Special Weight Structures*. *Algorithmica* 70(3), pp. 457–492, doi:10.1007/s00453-013-9843-7. arXiv:1604.08234. Announced at ESA 2012.

[16] Carlo Comin & Romeo Rizzi (2017): *Improved Pseudo-polynomial Bound for the Value Problem and Optimal Strategy Synthesis in Mean Payoff Games*. *Algorithmica* 77(4), pp. 995–1021, doi:10.1007/s00453-016-0123-1.

[17] Dani Dorfman, Haim Kaplan, Robert E. Tarjan & Uri Zwick (2023): *Optimal Energetic Paths for Electric Cars*. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi & Grzegorz Herman, editors: *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, *LIPIcs* 274, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 42:1–42:17, doi:10.4230/LIPIcs.ESA.2023.42.

[18] Dani Dorfman, Haim Kaplan & Uri Zwick (2019): *A Faster Deterministic Exponential Time Algorithm for Energy Games and Mean Payoff Games*. In: *Proc. of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 132, pp. 114:1–114:14, doi:10.4230/LIPIcs.ICALP.2019.114.

[19] Andrzej Ehrenfeucht & Jan Mycielski (1979): *Positional strategies for mean payoff games*. *International Journal of Game Theory* 8(2), pp. 109–113, doi:10.1007/BF01768705.

[20] Nathanaël Fijalkow, Paweł Gawrychowski & Pierre Ohlmann (2020): *Value Iteration Using Universal Graphs and the Complexity of Mean Payoff Games*. In: *Proc. of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, 170, pp. 34:1–34:15, doi:10.4230/LIPIcs.MFCS.2020.34.

[21] Vladimir A. Gurvich, Alexander V. Karzanov & L. G. Khachivan (1988): *Cyclic games and an algorithm to find minimax cycle means in directed graphs*. *USSR Computational Mathematics and Mathematical Physics* 28(5), pp. 85–91, doi:10.1016/0041-5553(88)90012-2.

[22] Loïc Hélouët, Nicolas Markey & Ritam Raha (2019): *Reachability Games with Relaxed Energy Constraints*. In: *Proceedings Tenth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2019, Bordeaux, France, 2-3rd September 2019*, *EPTCS* 305, pp. 17–33, doi:10.4204/EPTCS.305.2.

[23] Donald B. Johnson (1977): *Efficient Algorithms for Shortest Paths in Sparse Networks*. *J. ACM* 24(1), pp. 1–13, doi:10.1145/321992.321993.

[24] Marcin Jurdziński (1998): *Deciding the winner in parity games is in UP$\cap$ co-UP*. *Information Processing Letters* 68(3), pp. 119–124, doi:10.1016/S0020-0190(98)00150-1.

[25] Tomasz Kociumaka & Adam Polak (2023): *Bellman-Ford Is Optimal for Shortest Hop-Bounded Paths*. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi & Grzegorz Herman, editors: *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, *LIPIcs* 274, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 72:1–72:10, doi:10.4230/LIPIcs.ESA.2023.72. arXiv:2211.07325.

[26] Donald A. Martin (1975): *Borel determinacy*. *Annals of Mathematics* 102(2), pp. 363–371, doi:10.2307/1971035.

[27] Piotr Sankowski (2005): *Shortest Paths in Matrix Multiplication Time*. In Gerth Stølting Brodal & Stefano Leonardi, editors: *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, *Lecture Notes in Computer Science* 3669, Springer, pp. 770–778, doi:10.1007/11561071_68.

[28] Robert E. Tarjan (1972): *Depth-First Search and Linear Graph Algorithms*. *SIAM J. Comput.* 1(2), pp. 146–160, doi:10.1137/0201010.

[29] Virginia Vassilevska Williams & R. Ryan Williams (2018): *Subcubic Equivalences Between Path, Matrix, and Triangle Problems*. *J. ACM* 65(5), pp. 27:1–27:38, doi:10.1145/3186893. Announced at FOCS 2010.

[30] Raphael Yuster & Uri Zwick (2005): *Answering distance queries in directed graphs using fast matrix multiplication*. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, IEEE Computer Society, pp. 389–396, doi:10.1109/SFCS.2005.20.

[31] Uri Zwick & Mike Paterson (1996): *The complexity of mean payoff games on graphs*. *Theoretical Computer Science* 158(1-2), pp. 343–359, doi:10.1016/0304-3975(95)00188-3.