Polynomial-Time Algorithms for Energy Games with Special Weight Structures

Krishnendu Chatterjee[1,⋆], Monika Henzinger[2,⋆⋆], Sebastian Krinninger[2,⋆⋆], and Danupon Nanongkai[3,⋆⋆⋆]

[1] Institute of Science and Technology, Klosterneuburg, Austria
[2] University of Vienna, Faculty of Computer Science, Vienna, Austria
[3] Nanyang Technological University, Singapore, Singapore

Abstract. Energy games belong to a class of *turn-based two-player infinite-duration games* played on a weighted directed graph. It is one of the rare and intriguing combinatorial problems that lie in NP ∩ co-NP, but are not known to be in P. The existence of polynomial-time algorithms has been a major open problem for decades and apart from pseudopolynomial algorithms there is no algorithm that solves any non-trivial subclass in polynomial time.

In this paper, we give several results based on the weight structures of the graph. First, we identify a notion of *penalty* and present a polynomial-time algorithm when the penalty is large. Our algorithm is the first polynomial-time algorithm on a large class of weighted graphs. It includes several worst-case instances on which previous algorithms, such as value iteration and random facet algorithms, require at least subexponential time. Our main technique is developing the first non-trivial *approximation* algorithm and showing how to convert it to an exact algorithm. Moreover, we show that in a practical case in verification where weights are clustered around a constant number of values, the energy game problem can be solved in polynomial time. We also show that the problem is still as hard as in general when the clique-width is bounded or the graph is strongly ergodic, suggesting that restricting the graph structure does not necessarily help.

1 Introduction

Consider a coffee shop A having a budget of e competing with its rival B across the street who has an unlimited budget. Each competitor can set the price of a cup of coffee between 1 cent and 10 euros (as an integer cent amount). Coffee

⋆ Supported by the Austrian Science Fund (FWF): P23499-N23, the Austrian Science Fund (FWF): S11407-N23 (RiSE), an ERC Start Grant (279307: Graph Games), and a Microsoft Faculty Fellows Award.

⋆⋆ Supported by the Austrian Science Fund (FWF): P23499-N23, the Vienna Science and Technology Fund (WWTF) grant ICT10-002, the University of Vienna (IK I049-N), and a Google Faculty Research Award.

⋆⋆⋆ Work partially done while at University of Vienna, Austria.

shop B can observe the price of a cup of coffee at A, say p_0, and responds with a price p_1, causing A a loss of $w(p_0, p_1)$, which could potentially put A out of business. If A manages to survive, then it can respond to B with a price p_2, gaining itself a profit of $w(p_1, p_2)$. Then B will try to put A out of business again with a price p_3. How much initial budget e does A need in order to guarantee that its business will survive forever? This is a simple example of a perfect-information turn-based infinite-duration game called an *energy game*, defined as follows.

In an energy game, there are two players, Alice and Bob, playing a game on a finite directed graph $G = (V, E)$ with weight function $w : E \to \mathbb{Z}$. Each node in G belongs to either Alice or Bob. The game starts by placing an imaginary car on a specified starting node v_0 with an initial energy $e_0 \in \mathbb{Z}^{\geq 0} \cup \{\infty\}$ in the car (where $\mathbb{Z}^{\geq 0} = \{0, 1, \ldots\}$). The game is played in *rounds*: at any round $i > 0$, if the car is at node v_{i-1} and has energy e_{i-1}, then the owner of v_{i-1} moves the car from v_{i-1} to a node v_i along an edge $(v_{i-1}, v_i) \in E$. The energy of the car is then updated to $e_i = e_{i-1} + w(v_{i-1}, v_i)$. The goal of Alice is to sustain the energy of the car while Bob will try to make Alice fail. That is, we say that Alice *wins* the game if the energy of the car is never below zero, i.e. $e_i \geq 0$ for all i; otherwise, Bob wins. The problem of *computing the minimal sufficient energy* is to compute the minimal initial energy e_0 such that Alice wins the game. (Note that such e_0 always exists since it could be ∞ in the worst case.) Figure 1 shows an example run of an energy game. The important parameters in terms of running time are the number n of nodes in the graph, the number m of edges in the graph, and the weight parameter W defined as $W = \max_{(u,v)\in E} |w(u, v)|$.

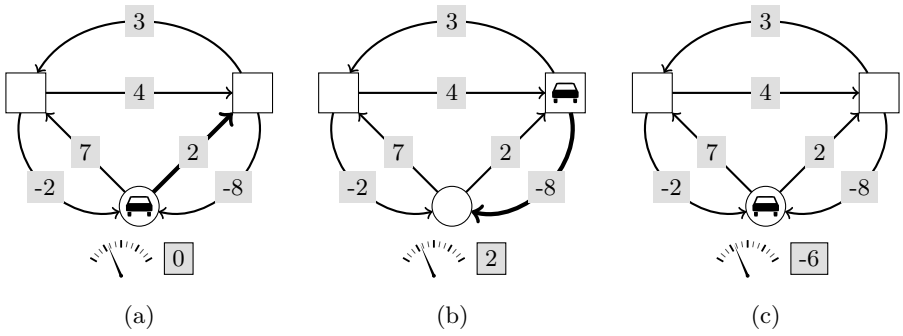

Fig. 1. An example of an energy game. The round node belongs to Alice and the rectangular nodes belong to Bob. The current energy level of the car is written in the box at the bottom. The game starts at the bottom node, which belongs to Alice, with the initial energy level 0 (a). Alice chooses to move the car to the upper right node using the edge of weight 2. Afterwards the car has energy 2 and is located on Bob's node (b). Bob chooses to move the car to the bottom node using the edge of weight -8. This decreases the energy of the car to -6 (c). At this point Alice has lost the game because the energy of the car is negative.

Related Work. Energy games belong to an intriguing family of *infinite-duration turn-based games* which includes alternating games [30], and has applications in areas such as computer-aided verification and automata theory [9, 3, 8], as well as in online and streaming problems [33]. Energy games are polynomial-time equivalent to *mean-payoff games* [6]. Furthermore there are polynomial-time reductions from *parity games* to energy games [20] and from energy games to *simple stochastic games* [10, 33].[4] These games are among the rare combinatorial problems, along with *Graph Isomorphism*, that are unlikely to be NP-complete (since they are in UP∩co-UP ⊆ NP∩co-NP [13, 18, 33, 20]) but not known to be in P. It is a major open problem whether any of these games are in P or not. While the energy game is relatively new and interesting in its own right, it has been implicitly studied since the late 80s, due to its close connection with the mean-payoff game. In particular, the seminal paper by Gurvich et al. [18] presents a simplex-like algorithm for mean-payoff games which computes a "potential function" that is essentially the energy function.[5]

The algorithm of Gurvich et al. [18] was shown to be pseudopolynomial by Pisaruk [29]. Another pseudopolynomial algorithm was given by Zwick and Paterson [33]. Björklund and Vorobyov [2] developed an algorithm for mean-payoff games that besides being pseudopolynomial has a randomized strongly subexponential running time of $2^{O(\sqrt{n \log n})} \log W$. Lifshits and Pavlov [25] described an exponential algorithm for mean-payoff games. Recently, Brim et al. [7] gave an algorithm for energy games that is faster than previous deterministic pseudopolynomial algorithms and runs in time $O(mnW)$. It yields the current fastest pseudopolynomial complexity for energy games as well as mean-payoff games. To the best of our knowledge only two special cases of energy or mean-payoff games are known to admit a polynomial-time algorithm. The first special case is where all nodes belong to one player and for example can be solved with Karp's minimum cycle mean algorithm [23]. The second special case is when W is polynomial in the input size and thus pseudopolynomial algorithms actually run in polynomial time.

Infinite-duration turn-based games also have strong connections to *(mixed) Nash equilibrium computation* [12] and *Linear Programming* [32]. For example, they are in a low complexity class lying very close to P called CCLS [12] which is in PPAD ∩ PLS. This implies that, unlike many problems in Game Theory, these games are *unlikely* to be PPAD-complete. Moreover, as shown by Halman [19], all these games are *LP-type problems* [31], a concept that generalizes linear programming. Therefore the random facet algorithm [21, 22, 27], a simplex-algorithm with a certain randomized pivoting rule, can be used to solve them

[4] Both reductions were originally shown for mean-payoff games.

[5] More precisely, the auxiliary algorithm of Gurvich et al. [18] solves a decision version of mean-payoff games. It has to output, for every node v, whether the mean-payoff at v is at least zero or not. If it is, the potential of v computed by this algorithm is equal to what we call the minimal sufficient energy of v. If not, we know that the minimal sufficient energy of v is ∞.

in randomized subexponential time[6]. This relates infinite-duration turn-based games to the question whether there exists a pivoting rule for the simplex algorithm that requires a polynomial number of pivoting steps on any linear program, which is perhaps one of the most important problems in the field of linear programming. In fact, several randomized pivoting rules have been conjectured to solve linear programs in polynomial time until recent breakthrough results [16, 14, 15] have rejected these conjectures. As noted by Friedmann et al. [16], infinite-duration turn-based games played an important role in this breakthrough as the lower bounds were first developed for these games and later extended to linear programs via Markov Decision Processes. Figure 2 summarizes the complexity status of energy games and related problems.

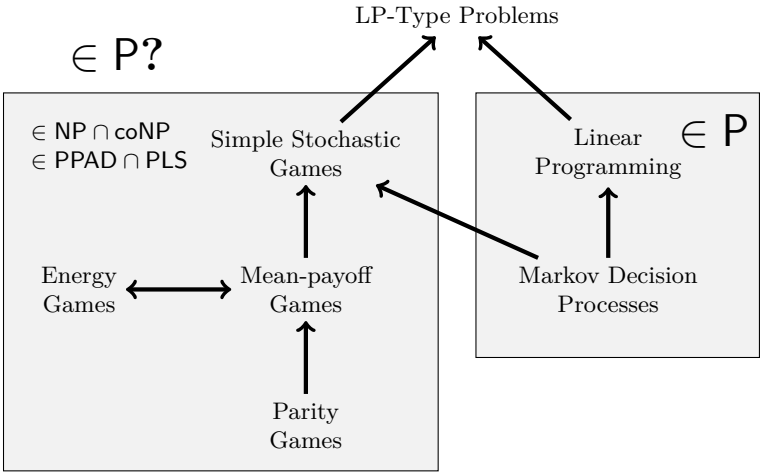

Fig. 2. The complexity status of energy games and related problems. Arrows indicate polynomial-time reductions.

Our Contributions. In this paper we identify several classes of graphs (based on weight structures) for which energy games can be solved in polynomial time. For any starting node s, let $e^*_{G,w}(s)$ denote the minimal sufficient energy. Our first contribution is an algorithm whose running time is based on a parameter called *penalty*. Informally, a penalty[7] of D means that Bob has a way to play optimally such that, for all choices of Alice, one of the following two situations occurs. (1) Alice wins the game for some finite initial energy. (2) Alice loses the game even if an additional energy of D would be added to the car in every turn. We denote the *penalty of the graph* (G, w) by $P(G, w)$. We show that the *higher* the penalty is, the faster we can compute the minimal energies.

[6] The randomized subexponential algorithm of Björklund and Vorobyov [2] uses the same randomization scheme as the random facet algorithm.

[7] We formally define the concept of penalty in Section 2.

Theorem 1. *Given a graph (G, w) and an integer M we can compute the minimal initial energies of all nodes in*

$$O\left(mn\left(\log\frac{M}{n}\right)\left(\log\frac{M}{n\lceil P(G,w)\rceil}\right)+m\frac{M}{\lceil P(G,w)\rceil}\right)$$

time, provided that for all v, $e^*_{G,w}(v) < \infty$ implies that $e^*_{G,w}(v) \leq M$.

We note that in addition to (G, w), our algorithm takes M as an input. If M is unknown, we can simply use the universal upper bound $M = nW$ [7]. Allowing different values of M will be useful in our proofs. We emphasize that the algorithm can run without knowing $P(G, w)$. Our algorithm is as efficient as the fastest known pseudopolynomial-time ($O(mnW)$-time) algorithm [7] in the general case where $M = nW$ and $P(G, w) = 1/n$ (so $\lceil P(G, w)\rceil = 1$).

If the penalty is at least $W/\operatorname{poly}(n)$, our algorithm runs in polynomial time. Therefore, the algorithm also solves several classes of graphs that are previously not known to be solvable in polynomial time. As an illustration, consider the class of graphs where each cycle has total weight either positive or less than $-W/2$. In this case, our algorithm runs in polynomial time. *All known worst-case instances* [18, 1, 33, 15] of previous algorithms fall in this class of graphs. In particular, we observe that, for this class of graphs, the following algorithms need at least subexponential time (while our algorithm runs in polynomial time): the algorithm by Gurvich et al. [18], the algorithm by Brim et al. [7], the algorithm by Zwick and Paterson [33] and the random facet algorithm by Matoušek et al. [27] (the latter two algorithms are used for the decision versions of mean-payoff and parity games, respectively).[8]

Our result might also be of a practical interest since it solves energy games faster when penalties are high while it runs with the same running time as previous pseudopolynomial-time algorithms [7] in the worst case.

Our second contribution is an algorithm that approximates the minimal energy within some *additive error* where the size of the error depends on the penalty. This result is the main tool in proving Theorem 1 where we show how to use the approximation algorithm to compute the minimal energy *exactly*.

Theorem 2. *Given a graph (G, w) with $P(G, w) \geq 1$, an integer M, and an integer c such that $n \leq c \leq nP(G, w)$, we can compute an energy function e such that $e(v) \leq e^*_{G,w}(v) \leq e(v) + c$ for every node v in $O(mnM/c)$ time, provided that for every node v, $e^*_{G,w}(v) < \infty$ implies that $e^*_{G,w}(v) \leq M$.*

[8] A worst-case instance for the first two algorithms has been developed by Lebedev and is mentioned by Gurvich et al. [18] and shown by Beffara and Vorobyov [1] (for the second algorithm, we exploit the fact that it is deterministic and there exists a bad ordering in which the nodes are processed). Worst-case instances for the third and the fourth algorithm have been given by Zwick and Paterson [33] and Friedmann et al. [15], respectively. We note that the instances shown by Beffara and Vorobyov [1] and Friedmann et al. [15] contain one cycle of small negative weight. One can change the value of this cycle to $-W$ to make these examples belong to the desired class of graphs without changing the worst-case behaviors of the mentioned algorithms.

The main technique in proving Theorem 2 is rounding weights appropriately. We note that a similar idea of approximation has been explored earlier in the case of mean-payoff games [5]. Roth et al. [30] show an *additive* FPTAS for rational weights in $[-1, 1]$. This implies an additive error of ϵW for any $\epsilon > 0$ in our setting. This does not help in general since the error depends on W. Boros et al. [5] later achieved a *multiplicative* error of $(1 + \epsilon)$. This result holds, however, only when the edge weights are non-negative integers. In fact, it is shown that if one can approximate the mean-payoff within a small multiplicative error in the general case, then the exact mean-payoff can be found [17]. Despite several results for mean-payoff games, there is currently *no* approximation algorithm for general energy games. Our algorithm is the first non-trivial approximation algorithm for the energy game.

Our third contribution is a variant of the *Value Iteration Algorithm* by Brim et al. [7] which runs faster in many cases. The running time of the algorithm depends on a concept that we call *admissible list* (defined in Section 3) which uses the weight structure. One consequence of this result is used to prove Theorem 2. The other consequence is an algorithm for what we call the *fixed-window* case.

Theorem 3. *If there are d values $w_1, \ldots, w_d$ and a window size δ such that for every edge $(u, v) \in G$ we have $w(u, v) \in \{w_i - \delta, \ldots, w_i + \delta\}$ for some $1 \leq i \leq d$, then the minimal energies can be computed in $O(m\delta n^{d+1})$ time.*

The fixed-window case, besides its theoretical attractiveness, is also interesting from a practical point of view. Energy and mean-payoff games have many applications in the area of verification, mainly in the synthesis of reactive systems with resource constraints [3] and performance aware program synthesis [8]. In most applications related to synthesis, the resource consumption is through only a few common operations, and each operation depending on the current state of the system consumes a related amount of resources. In other words, in these applications there are d groups of weights (one for each operation) where in each group the weights differ by at most δ (i.e, δ denotes the small variation in resource consumption for an operation depending on the current state), and d and δ are typically constant. Theorem 3 implies a polynomial-time algorithm for this case.

We also show that the energy game problem is still as hard as the general case even when the clique-width is bounded or the graph is strongly ergodic (see Section 6). This suggests that restricting the graph structures might not help in solving the problem, which is in sharp contrast to the fact that parity games can be solved in polynomial time in these cases [28, 24].

Theorem 4. *The energy game problem on arbitrary graphs is polynomial-time equivalent to the energy game problem on graphs that have bounded clique-width as well as to the energy game problem on graphs that are strongly ergodic.*

2 Preliminaries

Figure 3 summarizes the notation introduced in this section.

$G = (V, E)$	Directed graph with nodes V and edges E in which every node has out-degree ≥ 1
V_A (V_B)	Set of nodes controlled by Alice (Bob). $V_A \cup V_B = V$ and $V_A \cap V_B = \emptyset$
n	Number of nodes in G, i.e., $n = \|V\|$
m	Number of edges in G, i.e., $m = \|E\|$
$w(u, v)$	Weight of edge (u, v)
$w(P)$	Total weight of a finite path P, sum of all edge weights on P
W	Maximum absolute edge weight, $W = \max_{(u,v)\in E} \|w(u, v)\|$
$e^*_{G,w}(v)$	Minimal energy at node v in weighted graph (G, w)
$P(G, w)$	Penalty of weighted graph (G, w)
σ (τ)	A strategy of Alice (Bob), i.e., a function that maps every node $u \in V_A$ ($u \in V_B$) to a neighboring node v such that $(u, v) \in E$
(σ, τ)	A pair of strategies where σ is a strategy of Alice and τ is a strategy of Bob
σ^* (τ^*)	An optimal strategy of Alice (Bob)
$G(\sigma, \tau)$	Restriction of G to pair of strategies (σ, τ)

Fig. 3. Overview of notation defined in Section 2

Energy Games. An energy game is played by two players, Alice and Bob. Its input instance consists of a finite weighted directed graph (G, w) where all nodes have out-degree at least one[9]. The set of nodes V is partitioned into V_A and V_B, which belong to Alice and Bob respectively, and every edge $(u, v) \in E$ has an integer weight $w(u, v) \in \{-W, \ldots, W\}$. It can be assumed without loss of generality that there are no self-loops.[10] Additionally, we are given a node s and an initial energy e_0. To formally define energy games, we need the notion of *strategies*. While general strategies can depend on the history of the game, it has been shown that we can assume that if a player wins a game, a *positional strategy* suffices to win [9, 6].[11] Therefore we only consider positional strategies. A positional strategy σ of Alice is a mapping from each node in V_A to one of its out-neighbors, i.e., for any $u \in V_A$, $\sigma(u) = v$ for some $(u, v) \in E$. This means that Alice sends the car to v every time it is at u. We define a positional strategy τ of Bob similarly. We simply use "strategy" instead of "positional strategy" in the rest of the paper.

[9] (G, w) is usually called a "game graph" in the literature. We will simply say "graph".

[10] If some node v has a self-loop (v, v) of weight $w(v, v)$, we can replace the self-loop as follows: we add an artificial node v' and two edges (v, v') and (v', v). The edges (v, v') and (v', v) both get the weight $w(v, v)$. This does not change the energy values nor the average weight of any cycle.

[11] Positional strategies are both *pure* and *memoryless*, i.e., they are deterministic and do not depend on the history of the game. The existence of optimal positional strategies in energy games follows immediately from the existence of optimal positional strategies in mean-payoff games [13] and the reduction of energy games to mean-payoff games [6].

A *pair of strategies* (σ, τ) consists of a strategy σ of Alice and τ of Bob. For any pair of strategies (σ, τ), we define $G(\sigma, \tau)$ to be the subgraph of G having only edges corresponding to the strategies σ and τ; i.e.,

$$G(\sigma, \tau) = (V, E') \quad \text{where} \quad E' = \{(u, \sigma(u)) \mid u \in V_A\} \cup \{(u, \tau(u)) \mid u \in V_B\}.$$

In $G(\sigma, \tau)$ every node has a unique out-edge.

Now, consider an energy game played by Alice and Bob starting at node s with initial energy e_0 using strategies σ and τ, respectively. We use $G(\sigma, \tau)$ to determine who wins the game as follows. For any i, let P_i be the (unique) directed path of length i in $G(\sigma, \tau)$ originating at s. Observe that P_i is exactly the path that the car will be moved along for i rounds, and the energy of the car after i rounds is $e_i = e_0 + w(P_i)$ where $w(P_i)$ is the sum of the edge weights in P_i. We say that *Bob wins* the game if there exists i such that $e_0 + w(P_i) < 0$ and *Alice wins* otherwise. Equivalently, we can determine who wins as follows. Let C be the (unique) cycle reachable by s in $G(\sigma, \tau)$, and let $w(C)$ be the sum of the edge weights in C. If $w(C) < 0$, then Bob wins; otherwise, Bob wins if and only if there exists a *simple* path P_i of some length i such that $e_0 + w(P_i) < 0$.

This leads to the following definition of the *minimal sufficient energy* at node s *corresponding to strategies* σ and τ, denoted by $e^*_{G(\sigma,\tau),w}(s)$: If $w(C) < 0$, then $e^*_{G(\sigma,\tau),w}(s) = \infty$; otherwise, $e^*_{G(\sigma,\tau),w}(s) = \max\{0, -\min w(P_i)\}$ where the minimization is over all *simple* paths P_i in $G(\sigma, \tau)$ originating at s. We then define the *minimal sufficient energy* at node s to be

$$e^*_{G,w}(s) = \min_{\sigma} \max_{\tau} e^*_{G(\sigma,\tau),w}(s) \tag{1}$$

where the minimization and the maximization are over all positional strategies σ of Alice and τ of Bob, respectively. We note that it follows from Martin's determinacy theorem [26] that $\min_\sigma \max_\tau e^*_{G(\sigma,\tau),w}(s) = \max_\tau \min_\sigma e^*_{G(\sigma,\tau),w}(s)$, and thus it does not matter which player picks the strategy first. We say that a strategy σ^* of Alice is an *optimal strategy* if for any strategy τ of Bob, $e^*_{G(\sigma^*,\tau),w}(s) \leq e^*_{G,w}(s)$. Similarly, τ^* is an optimal strategy of Bob if for any strategy σ of Alice, $e^*_{G(\sigma,\tau^*),w}(s) \geq e^*_{G,w}(s)$.

We call any $e : V \to \mathbb{Z}^{\geq 0} \cup \{\infty\}$ an *energy function*. We call $e^*_{G,w}$ in Eq. (1) a *minimal sufficient energy function* or simply a *minimal energy function*. By this definition the minimal energy function is unique. If $e(s) \geq e^*_{G,w}(s)$ for all s, then we say that e is a *sufficient* energy function. The goal of the energy game problem is to compute $e^*_{G,w}$.

We say that a natural number M is an *upper bound on the finite minimal energy* if for every node v either $e^*_{G,w}(v) = \infty$ or $e^*_{G,w}(v) \leq M$. This means that every finite minimal energy is bounded from above by M. A universal upper bound is $M = nW$ [7].

Penalty. Let (G, w) be a weighted graph. For any node s and real $D \geq 0$, we say that s has a *penalty of at least* D if there exists an optimal strategy τ^* of Bob such that for any strategy σ of Alice, the following condition holds for the (unique) cycle C reachable by s in $G(\sigma, \tau^*)$: if $w(C) < 0$, then the average

weight on C is at most $-D$, i.e. $\sum_{(u,v)\in C} w(u,v)/|C| \leq -D$. See Figure 4 for an example.

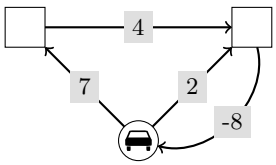

Fig. 4. The graph of this picture is the modification of the graph in Figure 1 where we have fixed Bob's optimal strategy. The round node belongs to Alice and the rectangular nodes belong to Bob. Alice has two strategies at the bottom node. If she chooses to go left, then the car runs into a cycle of total weight 3 and average weight 1. If she goes right, the car runs into a cycle of total weight -6 and average weight -3. Therefore the graph has penalty 3.

Intuitively, this means that either Alice wins the game using a finite initial energy, or she loses *significantly*, i.e., even if she would constantly receive an extra energy of a little less than D per round, she still needs an infinite initial energy in order to win the game. We note that $\sum_{(u,v)\in C} w(u,v)/|C|$ is known in the literature as the *mean-payoff* of s when Alice and Bob play according to σ and τ^*, respectively. Thus, the condition above is equivalent to saying that either the mean-payoff of s (when (σ,τ^*) is played) is non-negative or otherwise it is at most $-D$.

We define the penalty of s, denoted by $P_{G,w}(s)$, as the supremum[12] of all D such that s has a penalty of at least D. We say that the graph (G,w) has a penalty of at least D if every node s has a penalty of at least D, and define $P(G,w) = \min_{s\in G} P_{G,w}(s)$. Note that for any graph (G,w), $P(G,w) \geq 1/n$ since for any cycle C, $\sum_{(u,v)\in C} w(u,v)/|C|$ is either non-negative or at most $-1/n$.

3 Value Iteration Algorithm with Admissible List

In this section we present a variant of the *Value Iteration Algorithm* for computing the minimal energies of Brim et al. [7]. In addition to the graph (G,w), our algorithm uses one more parameter A which is a sorted list containing all possible minimal energy values. That is, the algorithm is promised that $e^*_{G,w}(v) \in A$ for every node v. We call any sorted list A such that $e^*_{G,w}(v) \in A$ for every node v an *admissible list*. We show the following proposition.

Proposition 5. *There is an algorithm that, given a (sorted) admissible list A, computes the minimal energies of all nodes in (G,w) in $O(m|A|)$ time.*

[12] We need to take the supremum here to include the case that s has penalty of at least D for every real D. In this case, $P_{G,w}(s) = \infty$ ($P_{G,w}(s)$ will not be well-defined if we use maximum instead of supremum).

In general, the simplest choice of an admissible list is $A = \{0, 1, \ldots, nW, \infty\}$. In this case the algorithm works like the current fastest pseudopolynomial algorithm by Brim et al. [7] and has a running time of $O(mnW)$. However, for some natural cases, we can give smaller admissible lists. Our first example are graphs where every edge weight is a multiple of an integer $B > 0$, as shown in the following corollary. This corollary will be used later in this paper.

Corollary 6. *Let (G, w) be a graph for which there is an integer $B > 0$ such that the weight of every edge $(u, v) \in G$ is of the form $w(u, v) = iB$ for some integer i, and M is an upper bound on the finite minimal energy (i.e., for any node v, if $e^*_{G,w}(v) < \infty$, then $e^*_{G,w}(v) \leq M$). There is an admissible list of size $O(M/B)$ which can be computed in $O(M/B)$ time. Thus there is an algorithm that computes the minimal energies of (G, w) in $O(mM/B)$ time.*

Our second example are graphs in which we have a (small) set of values $\{w_1, \ldots, w_d\}$ of size d and a window size δ such that *every* weight lies in $\{w_i - \delta, \ldots, w_i + \delta\}$ for one of the values w_i. This is exactly the situation described in Theorem 3. Since we prove this theorem in this section, we restate it here.

Theorem 3 (Restated). *If there are d values $w_1, \ldots, w_d$ and a window size δ such that for every edge $(u, v) \in G$ we have $w(u, v) \in \{w_i - \delta, \ldots, w_i + \delta\}$ for some $1 \leq i \leq d$, then the minimal energies can be computed in $O(m\delta n^{d+1})$ time.*

As noted in Section 1, in some applications d is a constant and δ is polynomial in n. In this case Theorem 3 implies a polynomial-time algorithm.

In the rest of this section we first give a proof of Proposition 5 (cf. Section 3.1). We subsequently use it to prove Corollary 6, and Theorem 3 (cf. Section 3.2). In both cases, we first highlight the main ideas before giving the full proofs.

3.1 Proof of Proposition 5

In the following we describe the modified value iteration algorithm for computing minimal energies and prove its correctness and running time as stated in Proposition 5. The value iteration algorithm relies on the following characterization of the minimal energy.

Lemma 7 (Minimal Energy Characterization [7]). *An energy function e is the minimal energy function of a weighted graph (G, w) if and only if it fulfills the following three conditions:*

1. *For every node $u \in V_A$, $e(u) + w(u, v) \geq e(v)$ for* some *edge $(u, v) \in E$.*
2. *For every node $u \in V_B$, $e(u) + w(u, v) \geq e(v)$ for* every *edge $(u, v) \in E$.*
3. *For every energy function e' that fulfills conditions 1 and 2 we have $e(v) \leq e'(v)$ for every node $v \in V$.*

Note that the first two conditions of this lemma are trivially satisfied for a node u if we set $e(u) = \infty$. An intuitive interpretation of the first two conditions is this: Consider any node u of Alice. If we believe that $e(v)$ is sufficient for

all neighbors v of u, then $e(u)$ should be sufficient if, when the car has energy $e(u)$ at u, she can move the car to some neighboring node v to make sure that the energy of the car is still sufficient, i.e., $e(u) + w(u, v) \geq e(v)$. Similarly, if u is Bob's node and we believe that $e(v)$ is sufficient for all neighbors v of u, then $e(u)$ should be sufficient if, when the car has energy $e(u)$ at u, it can be guaranteed that the energy is still sufficient for any neighbor v the car is moved to, i.e., $e(u) + w(u, v) \geq e(v)$ for all v.

The first two conditions give a sufficient condition for an energy function to be sufficient. It can be shown that these conditions are *not* necessary (i.e., some sufficient energy functions do not satisfy these conditions). However, an interesting property of these conditions is that it *is* necessary for an energy to be minimal. Since there could be non-minimal energy functions that satisfy the first two conditions, we have to add the third condition: at *all* nodes, the minimal energy function has to be smaller than *all* other functions that satisfy the first two conditions. All three conditions together characterize the (unique) minimal energy function.

We will first give a general algorithm based on value iteration, called Algorithm 1, in which nodes are "updated" in an arbitrary order. We will prove the correctness of this algorithm. Then we will present a second, faster algorithm, called Algorithm 2, that processes the nodes in a specific order and that uses a simple data structure. We will argue that this algorithm gives the desired running time.

The basic idea of Algorithm 1 is as follows. The algorithm starts with an energy function $e(v) = \min A$ for every node v and keeps increasing e slightly in an attempt to satisfy the first two conditions in Lemma 7. That is, as long as these conditions are not fulfilled for some node u, it increases $e(u)$ to the next value in A, which could also be ∞. This updating process is repeated until e satisfies the conditions (which will eventually happen at least when all $e(u)$ become ∞). This updating process of the algorithm is the same as in the algorithm of Brim et al. except that $e(u)$ always increases to the next value in A and not only to the value given by Lemma 7.

Correctness. Algorithm 1 shows a simplified version of the algorithm. We adapt the correctness proof of Brim et al. [7] to our notation. It turns out that our modification of the algorithm using a list of admissible values does not disturb the overall correctness argument.

We first prove the following invariant: after every iteration of the algorithm we have $e(x) \leq e^*_{G,w}(x)$ for every node x. The statement is certainly true before the first iteration: Since $e^*_{G,w}(x) \in A$ we have $\min A \leq e^*_{G,w}(x)$.

Now assume that $e(x) \leq e^*_{G,w}(x)$ for every node x at the beginning of the current iteration. Let u be the node that is updated in the current iteration. For every node $x \neq u$ the value of $e(x)$ does not change in the current iteration. Let $e'(u)$ be the value *before* the energy of u is increased to the next admissible value in Line 8 and let $e''(u)$ be the value after this operation. Since $e''(u) = \min\{r \in A \mid r \geq e'(u)\}$, it is sufficient to show that $e'(u) \leq e^*_{G,w}(u)$.

Algorithm 1: Modified value iteration algorithm

Input: A weighted graph (G, w), a sorted list A of admissible values for the minimal energies

Output: The minimal energy of (G, w)

```
1 e(u) ← min A for every u ∈ V                                  // Initialization
  // Repeat as long as some node u violates the first two conditions
     of Lemma 7
2 while there is a node u ∈ V such that u ∈ V_A and
        (u ∈ V_A and ∀(u,v) ∈ E : e(u) + w(u,v) < e(v)) or
        (u ∈ V_B and ∃(u,v) ∈ E : e(u) + w(u,v) < e(v)) do
      // Update e(u)
3     if u ∈ V_A then
4       e(u) ← min_{(u,v)∈E}(e(v) − w(u,v))
5     else if u ∈ V_B then
6       e(u) ← max_{(u,v)∈E}(e(v) − w(u,v))
      // Increase e(u) to next admissible value
7     e(u) ← min{r ∈ A | r ≥ e(u)}
8 return e
```

Consider first the case that $u \in V_A$. In this case we have $e(u)+w(u,y) < e(y)$ for every edge (u, y) because otherwise the algorithm would not update u. After the update (and before the execution of Line 8) we still have $e'(u) + w(u, y) \leq e(y)$ for every edge (u, y). Since $e^*_{G,w}$ is the minimal energy function we have $e^*_{G,w}(u) + w(u, v) \geq e^*_{G,w}(v)$ for some edge (u, v) by Lemma 7. By the induction hypothesis we have $e(v) \leq e^*_{G,w}(v)$.[13] Therefore we get

$$e'(u) + w(u, v) \leq e(v) \leq e^*_{G,w}(v) \leq e^*_{G,w}(u) + w(u, v)$$

and it follows that $e'(u) \leq e^*_{G,w}(u)$.

Consider now the case that $u \in V_B$. In this case we have $e(u)+w(u,v) < e(v)$ for at least one edge (u, v). After the update (and before the execution of Line 8) we still have $e'(u) + w(u, v) = e(v)$ for at least one edge (u, v). Since $e^*_{G,w}$ is the minimal energy function we have $e^*_{G,w}(u) + w(u, y) \geq e^*_{G,w}(y)$ for every edge (u, y) by Lemma 7. In particular this holds for the edge (u, v). By the induction hypothesis we have $e'(v) = e(v) \leq e^*_{G,w}(v)$. In total we get

$$e'(u) + w(u, v) = e(v) \leq e^*_{G,w}(v) \leq e^*_{G,w}(u) + w(u, v)$$

and it follows that $e'(u) \leq e^*_{G,w}(u)$.

This concludes the proof that for the energy function e returned by our algorithm we have $e(x) \leq e^*_{G,w}(x)$ for every node x. Clearly, the energy function returned by our algorithm fulfills the first two conditions of Lemma 7 because otherwise it would not have terminated. Thus, our algorithm returns the minimal

[13] Remember that we assume that there are no self-loops and therefore $v \neq u$.

energies, i.e., $e(v) = e_{G,w}(v)$ for every node v. We remark that the order in which the nodes are processed in the while loop is irrelevant for the correctness proof. We will use this fact in the following improved algorithm, Algorithm 2.

Running Time. A running time of $O(mn|A|)$ for Algorithm 1 is immediate as every node has to be updated at most $|A|$ times and both updating a node and checking whether it has to be updated takes time proportional to its out-degree. The speed-up technique of Brim et al. [7] also works for our modification and gives a running time of $O(m|A|)$. The idea is to maintain a counter for Alice's nodes that keeps track of the number of outgoing edges which fulfill the first condition of Lemma 7. The energy only has to be updated if the counter reaches 0. Algorithm 2 is the full algorithm which we show for the sake of completeness.

To show the correctness of this algorithm the following two invariants are needed:

1. For every node $u \in V \setminus L$ the following holds:
   - If $u \in V_A$, then there is an edge (u, v) such that $e(u) + w(u, v) \geq e(v)$
   - If $u \in V_B$, then for every edge (u, v) we have $e(u) + w(u, v) \geq e(v)$
2. If $u \in V_A \setminus L$, then $\text{count}(u) = |\{v \in V \mid (u, v) \in E, e(u) + w(u, v) \geq e(v)\}|$

The proof of these invariants does not differ from the one given by Brim et al. [7] which is why we omit it here. The update mechanism in Lines 10 to 14 is the same as in Algorithm 1 which we already proved to be correct.

We now obtain the desired running time of Algorithm 1 as follows. For every node u, we let $\deg^+(u)$ and $\deg^-(u)$ denote its out-degree and in-degree, respectively. The initialization steps in Lines 1 to 5 of the algorithm need time $O(\deg^+(u))$ for every node u. Thus, the total initialization cost is $O(\sum_{u \in U} \deg^+(u)) = O(m)$. Each iteration of the while loop in which we update a node u needs time $O(\deg^+(u) + \deg^-(u))$. Since the energy of every node can increase at most $|A|$ times, the total running time of this Algorithm 2 is

$$O\left(\sum_{u \in V} (\deg^+(u) + \deg^-(u)) \cdot |A|\right) = O(m|A|)\,.$$

This completes the proof of Proposition 5. □

3.2 Proofs of Corollary 6 and Theorem 3

We now prove that in the two special cases described in Corollary 6 and Theorem 3, we can give explicit formulations of admissible lists. For both proofs we first characterize what values the minimal energy can assume, dependent on the set of edge weights and an upper bound M on the finite minimal energy. Specifically, we define

$$U_M = \{0, \ldots, M, \infty\}\,. \tag{2}$$

We denote the set of different weights of a graph (G, w) by

$$R_{G,w} = \{w(u, v) \mid (u, v) \in E\}. \tag{3}$$

Algorithm 2: Modified value iteration algorithm with speed-up technique

Input: A weighted graph (G, w), a sorted list A of admissible values for the minimal energies

Output: The minimal energy of (G, w)

```
// Initialization
```
$L \leftarrow \{u \in V_A \mid \forall (u,v) \in E : e(u) + w(u,v) < e(v)\}$
$L \leftarrow \{u \in V_B \mid \exists (u,v) \in E : e(u) + w(u,v) < e(v)\} \cup L$
$e(u) \leftarrow \min A$ for every $u \in V$
$\mathrm{count}(u) \leftarrow 0$ for every $u \in V_A \cap L$
$\mathrm{count}(u) \leftarrow |\{v \in V \mid (u,v) \in E, e(u) + w(u,v) \geq e(v)\}|$ for every $u \in V_A \setminus L$

```
// Repeat as long as some node u violates the first two conditions
   of Lemma 7
```
while $L \neq \emptyset$ **do**
  Pick $u \in L$
  $L \leftarrow L \setminus \{u\}$
  $e_{\mathrm{old}} \leftarrow e(u)$
  ```
  // Update node u
  ```
  if $u \in V_A$ **then**
    $e(u) \leftarrow \min_{(u,v) \in E}(e(v) - w(u,v))$
  else if $u \in V_B$ **then**
    $e(u) \leftarrow \max_{(u,v) \in E}(e(v) - w(u,v))$
  $e(u) \leftarrow \min\{r \in A \mid r \geq e(u)\}$
  if $u \in V_A$ **then**
    $\mathrm{count}(u) \leftarrow |\{v \in V \mid (u,v) \in E, e(u) + w(u,v) \geq e(v)\}|$
  ```
  // Check whether neighbors of u have to be updated
  ```
  foreach $t \in V$ *such that* $(t,u) \in E$ *and* $e(t) + w(t,u) < e(u)$ **do**
    if $t \in V_A$ **then**
      if $e(t) + w(t,u) \geq e_{old}$ **then**
        $\mathrm{count}(t) \leftarrow \mathrm{count}(t) - 1$
      if $\mathrm{count}(t) \leq 0$ **then**
        $L \leftarrow L \cup \{t\}$
    else if $t \in V_B$ **then**
      $L \leftarrow L \cup \{t\}$

return e

The set of all (negated) combinations of edge weights is defined as

$$C_{G,w} = \left\{ -\sum_{i=1}^{k} x_i \mid x_i \in R_{G,w} \text{ for all } i, 0 \leq k \leq n \right\} \cup \{\infty\}. \tag{4}$$

Our key observation is the following lemma.

Lemma 8. *For every graph (G, w) with an upper bound M on the finite minimal energy we have $e^*_{G,w}(v) \in C_{G,w} \cap U_M$ for every node $v \in V$.*

Proof. If $e^*_{G,w}(v) = \infty$ then we clearly have $e^*_{G,w}(v) \in C_{G,w} \cap U_M$. If $e^*_{G,w}(v) < \infty$ we have $e^*_{G,w}(v) \in U_M$ since M is an upper bound on the finite minimal energy. We still have to show that $e^*_{G,w}(v) \in C_{G,w}$.

Let (σ^*, τ^*) be a pair of optimal strategies. Since σ^* and τ^* are optimal we have $e^*_{G,w}(v) = e^*_{G(\sigma^*,\tau^*),w}(v) < \infty$. By the definition of the minimal energy (see Section 2) we have

$$e^*_{G(\sigma^*,\tau^*),w}(v) = \max\{0, -\min_P w(P)\}$$

where the minimization is over all simple paths in $G(\sigma^*, \tau^*)$ originating at v and $w(P)$ denotes the sum of the edge weights of the path P. If $e^*_{G,w}(v) = 0$ we have $e^*_{G,w}(v) \in C_{G,w}$ by setting $k = 0$ (the empty sum has value 0). Otherwise we have

$$e^*_{G,w}(v) = -\sum_{(x,y)\in P} w(x, y)$$

for some simple path P in $G(\sigma^*, \tau^*)$ originating at v. Since the length of P is at most n we have at most n edges on P which makes it clear that $e^*_{G,w}(v) \in C_{G,w}$. □

Proof of Corollary 6. We want to use the value iteration algorithm of Proposition 5 with the list

$$A = \left\{ i \cdot B \mid 0 \leq i \leq \left\lceil \frac{M}{B} \right\rceil \right\} \cup \{\infty\}.$$

It is clear that A has size $O(M/B)$ and can be generated in $O(M/B)$ time. Thus, we only have to show that A is admissible to apply Proposition 5.

We will now show that $C_{G,w} \cap U_M \subseteq A$ where $C_{G,w}$ and U_M are as in Lemma 8. Let $y \in C_{G,w} \cap U_M$. The set of different edge weights is $R_{G,w} \subseteq \{i \cdot B \mid -W/B \leq i \leq W/B\}$. Since $y \in C_{G,w}$ there is some k $(0 \leq k \leq n)$ such that

$$y = -\sum_{j=1}^{k} x_j$$

where $x_j \in R_{G,w}$ for every $1 \leq j \leq k$. Therefore there is an integer i_j for every $1 \leq j \leq k$ such that $x_j = i_j B$ and we get

$$y = -\sum_{j=1}^{k} i_j B = -B \sum_{j=1}^{k} i_j = -iB$$

for some integer i. Since $y \in U_M$ we have $0 \leq -iB \leq M$ and therefore $0 \leq -i \leq M/B \leq \lceil M/B \rceil$. Thus, $y = -iB \in A$ which proves $C_{G,w} \cap U_M \subseteq A$. Since $C_{G,w} \cap U_M$ is admissible by Lemma 8 also A is admissible, i.e., $e^*_{G,w}(v) \in A$ for every node v. This completes the proof of Corollary 6. □

Proof of Theorem 3. We want to use the value iteration algorithm of Proposition 5 with the list

$$A' = \left\{ x - \sum_{j=1}^{k} w_{i_j} \mid 1 \leq i_j \leq d, 0 \leq k \leq n, -n\delta \leq x \leq n\delta \right\} \cup \{\infty\}.$$

To show Theorem 3 we have to prove three things:

1. A' is an admissible list.
2. A' has size $O(\delta n^{d+1})$.
3. A sorted version of A' can be computed in $O(\delta n^{d+1} + dn^d \log n)$ time.

We will now show that $C_{G,w} \subseteq A'$ where $C_{G,w}$ is as in Lemma 8. Let $y \in C_{G,w}$. By the definition of $C_{G,w}$ there is some k ($0 \leq k \leq n$) such that there are k edge weights $x_1, \ldots, x_k \in R_{G,w}$ such that

$$y = -\sum_{j=1}^{k} x_j .$$

By the structure of $R_{G,w}$, the set of all edge weights, we have, for every $1 \leq j \leq k$, $x_j = w_{i_j} + \delta_j$ for some i_j and δ_j such that $1 \leq i_j \leq d$ and $-\delta \leq \delta_j \leq \delta$ which gives

$$y = -\sum_{j=1}^{k} (w_{i_j} + \delta_j)$$

Now observe that

$$-\sum_{j=1}^{k} (w_{i_j} + \delta_j) = -\sum_{j=1}^{k} w_{i_j} - \sum_{j=1}^{k} \delta_j = x - \sum_{j=1}^{k} w_{i_j} .$$

for some x such that $-n\delta \leq -k\delta \leq x \leq k\delta \leq n\delta$. Therefore $y \in A'$ which proves that $C_{G,w} \subseteq A'$. Since $C_{G,w}$ is admissible by Lemma 8, also A' is admissible.

We now consider the size of A'. We define

$$S = \left\{ -\sum_{j=1}^{k} w_{i_j} \mid 1 \leq i_j \leq d \text{ for all } j, 0 \leq k \leq n, \right\}$$

and get that

$$A' = \{y + x \mid y \in S, -n\delta \leq x \leq n\delta\} \cup \{\infty\}.$$

We now bound the size of S as follows. Each element of S is a sum of at most n numbers, each chosen from $\{w_1, \ldots, w_d\}$. Therefore, such an element is of the form $\sum_{i=1}^{d} n_i w_i$ where each n_i is chosen from $\{0, 1, \ldots, n\}$. Thus, the size of S is $O(n^d)$ and the size of A' is $O(\delta n^{d+1})$.

For the computation of A' we first compute S. Sorting S takes time $O(|S| \cdot \log |S|)$ which is $O(dn^d \log n)$. We iterate over every element $y \in S$ and generate every integer i in $[y - n\delta, y + n\delta]$. We append i to the list A' if it is larger than the current last element of the list. Since for every $y \in S$ the interval that we consider has the same "width" of $n\delta$, it can never happen that we generate an integer i that is smaller than the last element and does not yet occur in the list. Therefore A' is always sorted. This process takes time $O(|A'|) = O(\delta n^{d+1})$. In total it takes time $O(\delta n^{d+1} + dn^d \log n)$ to compute A'.

By Proposition 5 it takes time $O(m|A'|)$ to compute the minimal energies. When we add the construction time of A' we get a total running time of $O(\delta m n^{d+1} + dn^d \log n)$. Note that it is always possible to group the edge weights into $d = m$ groups such that every group contains only one edge weight. Therefore we may assume that $d \leq m$. In that case the first term dominates the second term which gives a total running time of $O(\delta m n^{d+1})$. This completes the proof of Theorem 3. □

4 Approximating Minimal Energies for Large Penalties

This section is devoted to proving Theorem 2. We restate it here for convenience.

Theorem 2 (Restated). *Given a graph (G, w) with $P(G, w) \geq 1$, an integer M, and an integer c such that $n \leq c \leq nP(G, w)$, we can compute an energy function e such that $e(v) \leq e^*_{G,w}(v) \leq e(v) + c$ for every node v in $O(mnM/c)$ time, provided that for every node v, $e^*_{G,w}(v) < \infty$ implies that $e^*_{G,w}(v) \leq M$.*

We show that we can approximate the minimal energy of nodes in high-penalty graphs (see Section 2 for the definition of penalty). The key idea is *rounding edge weights*, as follows. For an integer $B > 0$ we denote the weight function resulting from rounding up every edge weight to the nearest multiple of B by w_B. Formally, the function w_B is given by

$$w_B(u, v) = \left\lceil \frac{w(u, v)}{B} \right\rceil \cdot B$$

for every edge $(u, v) \in E$. Our algorithm is as follows. We set $B = \lfloor c/n \rfloor \leq P(G, w)$ (where c is as in Theorem 2). Since weights in (G, w_B) are multiples of B, e^*_{G,w_B} can be found faster than $e^*_{G,w}$ due to Corollary 6: we can compute e^*_{G,w_B} in time $O(mM/B) = O(mnM/c)$ provided that M is an upper bound on the finite minimal energy. This is the running time stated in Theorem 2. We complete the proof of Theorem 2 by showing that e^*_{G,w_B} is a good approximation of $e^*_{G,w}$ (i.e., it is the desired function e). Recall that by the definition of $P(G, w)$ every node v has penalty $P_{G,w}(v) \geq P(G, w)$.

Proposition 9. *For every node v with penalty $P_{G,w}(v) \geq B = \lfloor c/n \rfloor$ (where $c \geq n$) we have*

$$e^*_{G,w_B}(v) \overset{(1)}{\leq} e^*_{G,w}(v) \overset{(2)}{\leq} e^*_{G,w_B}(v) + nB \leq e^*_{G,w_B}(v) + c\,.$$

The rest of this section is devoted to proving Proposition 9. [14] Let us first give the proof ideas. The last inequality in the proposition follows immediately from the definition of B. The first two inequalities will be proved in Section 4.1 and 4.2. Let us first outline the proofs of these inequalities here. Inequality (1) is quite intuitive: We are doing Alice a favor by *increasing* edge weights from w to w_B. Thus, Alice should not require more energy in (G, w_B) than she needs in (G, w). As we show in Lemma 13 in Section 4.1, this actually holds for *any* increase in edge weights: For any w' such that $w'(u, v) \geq w(u, v)$ for all $(u, v) \in G$, we have $e^*_{G,w'}(v) \leq e^*_{G,w}(v)$. Thus we get the first inequality by setting $w' = w_B$.

For inequality (2) in Proposition 9, unlike the first inequality, we do not state this result for general increases of the edge weights as the bound depends on our rounding procedure. At this point we also need the precondition that the graph we consider has penalty at least B. We first show that the inequality holds when the strategies played by both players fulfill a certain condition, formally stated as follows (we prove this lemma in Section 4.2).

Lemma 10. *Let (σ, τ) be a pair of strategies. For any node v, if $e^*_{G(\sigma,\tau),w}(v) = \infty$ implies $e^*_{G(\sigma,\tau),w_B}(v) = \infty$, then $e^*_{G(\sigma,\tau),w}(v) \leq e^*_{G(\sigma,\tau),w_B}(v) + nB$.*

The above lemma needs a pair of strategies (σ, τ) such that $e^*_{G(\sigma,\tau),w}(v) = \infty$ implies $e^*_{G(\sigma,\tau),w_B}(v) = \infty$. This property can be explained as follows: If Alice needs infinite energy at node v in the graph $(G(\sigma, \tau), w)$ then she also needs infinite energy in the rounded-weight graph $(G(\sigma, \tau), w_B)$. Our second crucial fact shows that if v has penalty at least B then there exists a pair of strategies that has this property. *This is where we exploit the fact that the penalty is large.*

Lemma 11. *Let v be a node with penalty $P_{G,w}(v) \geq B$. Then there is an optimal strategy τ^* of Bob such that for every strategy σ of Alice we have that $e^*_{G(\sigma,\tau^*),w}(v) = \infty$ implies $e^*_{G(\sigma,\tau^*),w_B}(v) = \infty$.*

To prove Lemma 10 we only have to consider a special graph where the strategies of both players are fixed and thus all nodes have out-degree one. The challenge in proving Lemma 11 is to use the "right" strategy τ^*. We use the

[14] At this point we remark that energy games are not as resistant to perturbations of weights as mean-payoff games. In particular, if $w(u, v) \leq w'(u, v) \leq w(u, v) + x$ for every edge (u, v) and some positive constant x, then also $\mathrm{val}(v) \leq \mathrm{val}'(v) \leq \mathrm{val}(v)+x$, where $\mathrm{val}(v)$ and $\mathrm{val}'(v)$ are the values of the mean-payoff games for v in (G, w) and (G, w'), respectively. A similar inequality is not true for the minimal energies. Consider a cycle of total weight 0. By adding -1 to each edge weight, the weight of this cycle changes from non-negative to negative. Thus, the minimal energy might change from 0 to ∞.

strategy τ^* that comes from the definition of the penalty (cf. Section 2). The full proofs of Lemmas 10 and 11 are given in Section 4.2.

The other challenge of the proof of Proposition 9 is translating our result from graphs with fixed strategies to general graphs in order to prove the second inequality in Proposition 9. We do this as follows. Let σ^* be an optimal strategy of Alice for (G, w) and let (σ_B^*, τ_B^*) be a pair of optimal strategies for (G, w_B). Since v has penalty $P_{G,w}(v) \geq B$, Lemma 11 tells us that the preconditions of Lemma 10 are fulfilled. We use Lemma 10 and get that there is an optimal strategy τ^* of Bob such that $e^*_{G(\sigma_B^*,\tau^*),w}(v) \leq e^*_{G(\sigma_B^*,\tau^*),w_B}(v) + nB$. We now arrive at the chain of inequalities

$$\begin{aligned} e^*_{G,w}(v) &\overset{\text{(a)}}{=} e^*_{G(\sigma^*,\tau^*),w}(v) \overset{\text{(b)}}{\leq} e^*_{G(\sigma_B^*,\tau^*),w}(v) \overset{\text{(Lem. 10)}}{\leq} e^*_{G(\sigma_B^*,\tau^*),w_B}(v) + nB \\ &\overset{\text{(c)}}{\leq} e^*_{G(\sigma_B^*,\tau_B^*),w_B}(v) + nB \overset{\text{(d)}}{=} e^*_{G,w_B}(v) + nB \end{aligned}$$

that can be explained as follows. Since (σ^*, τ^*) and (σ_B^*, τ_B^*) are pairs of *optimal* strategies, we have (a) and (d). Due to the optimality we also have $e^*_{G(\sigma^*,\tau^*),w}(v) \leq e^*_{G(\sigma,\tau^*),w}(v)$ for *any* strategy σ of Alice, and in particular σ_B^*, which implies (b). A symmetric argument gives (c).

4.1 Proof of the First Inequality of Proposition 9

In the following we prove that an increase in edge weights does not increase the minimal energy for any node. We first prove the claim for the case where we fix the strategies of both players, i.e., on graphs where we have deleted all edges except those corresponding to the strategies of Alice and Bob. Afterwards we generalize the claim to arbitrary graphs.

Lemma 12. *Let G be a graph and w_1 and w_2 be edge weights such that $w_1(u, v) \leq w_2(u, v)$ for every edge $(u, v) \in G$. Then, for every pair of strategies (σ, τ) and every node $v \in G$, we have $e^*_{G(\sigma,\tau),w_1}(v) \geq e^*_{G(\sigma,\tau),w_2}(v)$.*

Proof. Let v be any node and (σ, τ) be any pair of strategies. First, consider the case where $e^*_{G(\sigma,\tau),w_2}(v) = \infty$. Let C denote the unique cycle reachable from v in $G(\sigma, \tau)$. Since $e^*_{G(\sigma,\tau),w_2}(v) = \infty$ we know by the definition of the minimal energy that $w_2(C) < 0$ where $w_2(C)$ denotes the sum of the edge weights of the cycle C. By our assumption we have $w_1(C) \leq w_2(C) < 0$, meaning that $e^*_{G(\sigma,\tau),w_1}(v) = \infty$ which is exactly what our inequality claims.

Next, consider the case where $e^*_{G(\sigma,\tau),w_2}(v) < \infty$. By the definition of the minimal energy (see Section 2) we have

$$e^*_{G(\sigma,\tau),w_2}(v) = \max\left\{0, -\min_P w_2(P)\right\}$$

where the minimization is over all simple paths in $(G(\sigma, \tau), w_2)$ originating at v and $w_2(P)$ denotes the sum of the edge weights of the path P.

In the case where $e^*_{G(\sigma,\tau),w_2}(v) = 0$, we have $e^*_{G(\sigma,\tau),w_2}(v) = 0 \leq e^*_{G(\sigma,\tau),w_1}(v)$. If $e^*_{G(\sigma,\tau),w_2}(v) > 0$, we have

$$e^*_{G(\sigma,\tau),w_2}(v) = -\min_P w_2(P)\,.$$

Since $w_2(P) \geq w_1(P)$ for every path P we have

$$\begin{aligned} e^*_{G(\sigma,\tau),w_2}(v) &= -\min_P w_2(P) \\ &\leq -\min_P w_1(P) \\ &\leq \max\{0, -\min_P w_1(P)\} \\ &= e^*_{G(\sigma,\tau),w_1}(v)\,. \end{aligned}$$

□

It is now straightforward to generalize the previous lemma by applying it to an *optimal* pair of strategies.

Lemma 13. *Let G be a graph and w_1 and w_2 be edge weights such that $w_1(u,v) \leq w_2(u,v)$ for every edge $(u,v) \in G$. Then $e^*_{G,w_1}(v) \geq e^*_{G,w_2}(v)$ for every node v.*

Proof. Let (σ_1^*, τ_1^*) be an optimal pair of strategies for (G, w_1) and let (σ_2^*, τ_2^*) be an optimal pair of strategies for (G, w_2). Note that $e^*_{G(\sigma_1^*,\tau_1^*),w_1}(v) \geq e^*_{G(\sigma_1^*,\tau),w_1}(v)$ for every strategy τ of Bob (since τ_1^* is Bob's optimal strategy). We also have $e^*_{G(\sigma,\tau_2^*),w_2}(v) \geq e^*_{G(\sigma_2^*,\tau_2^*),w_2}(v)$ for every strategy σ of Alice. Together with Lemma 12 we get

$$\begin{aligned} e^*_{G,w_1}(v) = e^*_{G(\sigma_1^*,\tau_1^*),w_1}(v) &\geq e^*_{G(\sigma_1^*,\tau_2^*),w_1}(v) \geq e^*_{G(\sigma_1^*,\tau_2^*),w_2}(v) \\ &\geq e^*_{G(\sigma_2^*,\tau_2^*),w_2}(v) = e^*_{G,w_2}(v)\,. \end{aligned}$$

□

4.2 Proof of the Second Inequality of Proposition 9

We now complete the proof of the second inequality of Proposition 9. We have already proved this inequality right after the statement of Proposition 9, but our proof assumes Lemmas 10 and 11. In this section, we provide the proofs of these two lemmas.

Lemma 10 (Restated). *Let (σ,τ) be a pair of strategies. For any node v, if $e^*_{G(\sigma,\tau),w}(v) = \infty$ implies that $e^*_{G(\sigma,\tau),w_B}(v) = \infty$, then $e^*_{G(\sigma,\tau),w}(v) \leq e^*_{G(\sigma,\tau),w_B}(v) + nB$.*

Proof. Recall that w_B is defined as the weight function resulting from rounding up every edge weight of w to the nearest multiple of B, i.e.,

$$w_B(u,v) = \left\lceil \frac{w(u,v)}{B} \right\rceil \cdot B\,.$$

By this definition we have $w_B(u,v) \leq w(u,v) + B$ for every edge $(u,v) \in E$.

If $e^*_{G(\sigma,\tau),w}(v) = \infty$, then also $e^*_{G(\sigma,\tau),w_B}(v) = \infty$ which trivially makes the inequality $e^*_{G(\sigma,\tau),w}(v) \leq e^*_{G(\sigma,\tau),w_B}(v) + nB$ hold. We now consider the case where $e^*_{G(\sigma,\tau),w}(v) < \infty$. By the definition of the minimal energy we have

$$e^*_{G(\sigma,\tau),w}(v) = \max\left\{0, -\min_P w(P)\right\}$$

where the minimization is over all simple paths in $(G(\sigma,\tau),w)$ originating at v and $w(P)$ denotes the sum of the edge weights of the path P. In the case where $e^*_{G(\sigma,\tau),w}(v) = 0$, our claimed inequality trivially holds because $e^*_{G(\sigma,\tau),w_B}(v) \geq 0$. Consider now the second case where $e^*_{G(\sigma,\tau),w}(v) > 0$. In this case, we have

$$e^*_{G(\sigma,\tau),w}(v) = -\min_P w(P)\,.$$

Every simple path P has length at most n and therefore

$$w_B(P) = \sum_{(u,v)\in P} w_B(u,v) \leq \sum_{(u,v)\in P} (w(u,v) + B) \leq w(P) + nB\,.$$

Thus, we get $w(P) \geq w_B(P) - nB$ for every simple path P. We now get

$$\begin{aligned} e^*_{G(\sigma,\tau),w}(v) = -\min_P w(P) &\leq -\min_P (w_B(P) - nB) \\ &= -\min_P (w_B(P)) + nB = e^*_{G(\sigma,\tau),w_B}(v) + nB\,. \end{aligned}$$

□

We now show that the precondition of the previous lemma is already implied by our choice of B.

Lemma 11 (Restated). *Let v be a node with penalty $P_{G,w}(v) \geq B$. Then there is an optimal strategy τ^* of Bob such that for every strategy σ of Alice we have that $e^*_{G(\sigma,\tau^*),w}(v) = \infty$ implies $e^*_{G(\sigma,\tau^*),w_B}(v) = \infty$.*

To prove the above lemma, we first prove the following claim.

Claim. If the average weight of a cycle C in (G,w) is at most $-B$, then C is a negative cycle in (G,w_B) with total weight $w_B(C) < 0$.

Proof. We assume that the average weight of C in (G,w) is at most $-B$, i.e.,

$$\frac{\sum_{(u,v)\in C} w(u,v)}{|C|} \leq -B\,.$$

Since $w_B(u,v) < w(u,v) + B$ for every edge $(u,v) \in E$, we get the following bound for the average weight of C in (G, w_B):

$$\begin{aligned}\frac{\sum_{(u,v)\in C} w_B(u,v)}{|C|} &< \frac{\sum_{(u,v)\in C}(w(u,v)+B)}{|C|} \\ &= \frac{\sum_{(u,v)\in C} w(u,v)}{|C|} + \frac{\sum_{(u,v)\in C} B}{|C|} \\ &\leq -B + \frac{|C|\cdot B}{|C|} = 0\,.\end{aligned}$$

Therefore, $w_B(C) = \sum_{(u,v)\in C} w_B(u,v) < 0$ which means that C is a negative cycle in (G, w_B). This finishes the proof of the claim. □

We now give the proof of Lemma 11.

Proof (Proof of Lemma 11). By the definition of the penalty we know that there is an optimal strategy τ^* of Bob such that, for every strategy σ of Alice, if the unique cycle C reachable from v in $G(\sigma, \tau^*)$ has negative total weight $w(C) < 0$, then its average weight is at most $-P_{G,w}(v) \leq -B$ by the definition of $P(G, w)$. Now let σ be any strategy of Alice and let C denote the unique cycle C reachable from v in $G(\sigma, \tau^*)$. Assume that $e^*_{G(\sigma,\tau^*),w}(v) = \infty$. Then we have $w(C) < 0$ and thus, by the definition of the penalty, C has an average weight of at most $-B$. By our claim we get that C is a negative cycle in (G, w_B) (i.e. $w_B(C) < 0$) and therefore $e^*_{G(\sigma,\tau^*),w_B}(v) = \infty$. □

5 Exact Solution by Approximation

We now use our results from the previous sections to prove Theorem 1.

Theorem 1 (Restated). *Given a graph (G, w) and an integer M we can compute the minimal initial energies of all nodes in*

$$O\left(mn\left(\log\frac{M}{n}\right)\left(\log\frac{M}{n\lceil P(G,w)\rceil}\right) + m\frac{M}{\lceil P(G,w)\rceil}\right)$$

time, provided that for all v, $e^*_{G,w}(v) < \infty$ implies that $e^*_{G,w}(v) \leq M$.

As the first step, we provide an algorithm that computes the minimal energy given a lower bound on the penalty of the graph. For this algorithm, we show how we can use the approximation algorithm in Section 4 to find an *exact* solution.

Lemma 14. *There is an algorithm that takes a graph (G, w), a lower bound D on the penalty $P(G, w)$, and an upper bound M on the finite minimal energy of (G, w) as its input and computes the minimal energies of (G, w) in $O(mn \log D + m \cdot \frac{M}{\lceil D\rceil})$ time. Specifically, if $P(G, w) \geq M/(2n)$, we can set $D = M/(2n)$ and the algorithm runs in time $O(mn \log(M/n))$.*

Proof (Main Idea). We provide the main idea of the proof of Lemma 14. Details are in Section 5.1 and 5.2.

To illustrate the main idea, we focus on the case $D = M/(2n)$ where we want to show an $O(mn \log(M/n))$ running time. If that condition does not hold, we can transform the problem into a problem where it holds in time $O(mM/D)$. Let $\mathcal{A}$ be the approximation algorithm given in Theorem 2. Recall that $\mathcal{A}$ takes c as its input and returns $e(v)$ such that $e(v) \leq e^*_{G,w}(v) \leq e(v) + c$ provided that $n \leq c \leq nP(G, w)$. Our exact algorithm will run $\mathcal{A}$ with parameter $c = \lfloor M/2 \rfloor$ which satisfies $c \leq M/2 \leq nD \leq nP(G, w)$. By Theorem 2, this takes $O(mnM/c) = O(mn)$ time. Using the energy function e returned by $\mathcal{A}$, our algorithm produces a new graph (G, w') defined by $w'(u, v) = w(u, v)+e(u)-e(v)$ for every edge (u, v) . It can be proved that this graph has the following crucial properties (see details in Lemma 16 in Section 5.1):

1. The penalty does not change, i.e., $P_{G,w}(v) = P_{G,w'}(v)$ for every node v.
2. We have $e^*_{G,w}(v) = e^*_{G,w'}(v) + e(v)$ for every node v.
3. The largest finite minimal energy of nodes in (G, w') is at most c; i.e., if $e^*_{G,w'}(v) < \infty$ then $e^*_{G,w'}(v) \leq c$. (This follows from property 2 and the inequality $e^*_{G,w}(v) \leq e(v) + c$ of Theorem 2.)

The algorithm then recurses on input (G, w'), D and $M' = c = \lfloor M/2 \rfloor$. Properties 1 and 3 guarantee that the preconditions of our algorithm for the recursive call are fulfilled: By our choice of M' we know that if $e^*_{G,w'}(v) < \infty$ then $e^*_{G,w'}(v) \leq M'$ and since $D \leq P_{G,w}(v) = P_{G,w'}(v)$, D is a lower bound on the penalty of (G, w'). Therefore we may recurse and the algorithm will return $e^*_{G,w'}(v)$ for every node v. It then outputs $e^*_{G,w'}(v) + e(v)$ which is guaranteed to be a correct solution (i.e., $e^*_{G,w}(v) = e^*_{G,w'}(v) + e(v)$) by the second property. The running time of this algorithm is $T(n, m, M) \leq T(n, m, M/2) + O(mn)$. We stop the recursion when M becomes small enough, i.e. when $M \leq n$. In this case the value iteration algorithm $\mathcal{A}$ runs in $O(mn)$ time. Thus we get $T(n, m, M) = O(mn \log(M/n))$ as desired. □

We now prove Theorem 1 by extending the algorithm of Lemma 14 to an algorithm that does not require the knowledge of a lower bound of the penalty.

Proof (Proof of Theorem 1). We repeatedly guess a lower bound for the penalty $P_{G,w}$ and run the algorithm of Lemma 14 until our guess eventually turns out to be correct. We start with the guess $D = M/(2n)$ for which the algorithm of Lemma 14 runs in time $O(mn \log(M/n))$. We then perform binary search for the next values of D by trying the values $M/(2n)$, $M/(4n)$, $M/(8n)$, and so on.

If our guess was correct, the algorithm returns the minimal energy function. If our guess was not correct, the energy function returned by our algorithm might not necessarily be the minimal energy function. Using the following characterization of the minimal energy we can check in linear time whether we have already found the minimal energy function.

Lemma 15 (Minimal Energy Characterization [25]). *The minimal energy of the graph (G, w) is the unique energy function e satisfying*

$$e(u) = \begin{cases} \min_{(u,v)\in E} \max(e(v) - w(u,v), 0) & \text{if } u \in V_A \\ \max_{(u,v)\in E} \max(e(v) - w(u,v), 0) & \text{if } u \in V_B \,. \end{cases}$$

for every node $u \in G$.

By checking the equation for every node u we can determine in time $O(m)$ whether an energy function e is indeed the minimal energy function.

We stop if we have already found the minimal energy function by running the algorithm of Lemma 14 with our guessed lower bound D of the penalty. Otherwise we guess a new lower bound D of the penalty which is half of the previous one and run the algorithm of Lemma 14 again. Eventually, our guess will be correct and we will stop before the guessed value is smaller than $P(G, w)/2$ or 1 (in the latter case we simply run the value iteration algorithm). Therefore we get a running time of

$$O\left(mn\left(\log \tfrac{M}{2n} + \log \tfrac{M}{4n} + \ldots + \log(\lceil P(G,w)\rceil)\right) + m\left(2n + 4n + \ldots + \tfrac{M}{\lceil P(G,w)\rceil}\right)\right)$$

which solves to $O(mn(\log \frac{M}{n})(\log \frac{M}{n\lceil P(G,w)\rceil}) + \frac{mM}{\lceil P(G,w)\rceil})$. □

In the worst case, i.e., when $P(G, w) = 1/n$ and $M = nW$, our algorithm runs in time $O(mnW)$ which matches the current fastest pseudopolynomial algorithm [7]. The result also implies that graphs with a penalty of at least $W/\operatorname{poly}(n)$ form an interesting class of polynomial-time solvable energy games.

5.1 Auxiliary Lemma Needed for Proving Lemma 14

In the following we prove an auxiliary lemma that we need for arguing about the correctness of the algorithm of Lemma 14. In that algorithm we first compute an energy function e that approximates the minimal energy function of a weighted graph (G, w) and then define a new weight function w' by $w'(u, v) = w(u, v) + e(u) - e(v)$ for every edge (u, v). For our algorithm to be correct we need two properties to hold.[15]

1. The penalty does not change, i.e., $P_{G,w}(v) = P_{G,w'}(v)$ for every node v.
2. We have $e^*_{G,w}(v) = e^*_{G,w'}(v) + e(v)$ for every node v.

We will show that these two properties actually hold for *any* energy function e.

Note that this kind of modification of the weights is often called a potential transformation [18] by the potential function e. It is well-known that a potential transformation does not change the average weight of any cycle and the total weight of a path from u to v changes by $e(u) - e(v)$. The first property above in

[15] The third property we mentioned above follows from property 2 and the approximation guarantee of the energy function e.

fact follows from this observation and we provide its proof only for completeness. The second property above additionally needs the precondition that $e(v)$ does not exceed the minimal energy at v and is not true for an arbitrary potential transformation.

Lemma 16. *Let (G, w) be a weighted graph and let e be an energy function such that $e(v) \leq e^*_{G,w}(v)$ for all $v \in G$. Define the modified game (G, w') with the weight function w' by $w'(u, v) = w(u, v) + e(u) - e(v)$ for every edge $(u, v) \in G$. Then the penalty does not change, i.e., $P_{G,w}(v) = P_{G,w'}(v)$ for every node $v \in G$, and $e^*_{G,w}(v) = e(v) + e^*_{G,w'}(v)$ for every node $v \in G$.*

Proof. We first show that the penalty does not change from w to w', i.e., $P_{G,w} = P_{G,w'}$. For this purpose we will show that every cycle in G has the same sum of edge weights in (G, w) and in (G, w') which means that the average weights are the same. By the definition of the penalty this implies that $P_{G,w}(v) = P_{G,w'}(v)$ for every node $v \in G$ as desired. Let C be a cycle of G consisting of the nodes $v_1, \ldots, v_k$. We simply plug in the definition of w' to check that our claim is true:

$$\begin{aligned}
\sum_{(u,v)\in C} w'(u, v) &= w'(v_k, v_1) + \sum_{i=1}^{k-1} w'(v_i, v_{i+1}) \\
&= w(v_k, v_1) + e(v_k) - e(v_1) + \sum_{i=1}^{k-1} (w(v_i, v_{i+1}) + e(v_i) - e(v_{i+1})) \\
&= w(v_k, v_1) + e(v_k) - e(v_1) + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) + \sum_{i=1}^{k-1} e(v_i) - \sum_{i=2}^{k} e(v_i) \\
&= w(v_k, v_1) + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) + \sum_{i=1}^{k} e(v_i) - \sum_{i=1}^{k} e(v_i) \\
&= w(v_k, v_1) + \sum_{i=1}^{k-1} w(v_i, v_{i+1}) \\
&= \sum_{(u,v)\in C} w(u, v)\,.
\end{aligned}$$

We now prove the second property. We define the energy function f by $f(v) = e(v) + e^*_{G,w'}(v)$ for every node $u \in G$. We use Lemma 15 to show that f is the minimal energy $e^*_{G,w}$. We have to show that, for every node $u \in G$, we have

$$f(u) = \begin{cases} \min_{(u,v)\in E} \max(f(v) - w(u, v), 0) & \text{if } u \in V_A \\ \max_{(u,v)\in E} \max(f(v) - w(u, v), 0) & \text{if } u \in V_B \end{cases}.$$

By the definition of f this is equivalent to

$$e(u) + e^*_{G,w'}(u) = \begin{cases} \min_{(u,v)\in E} \max(e^*_{G,w'}(v) - w(u, v) + e(v), 0) & \text{if } u \in V_A \\ \max_{(u,v)\in E} \max(e^*_{G,w'}(v) - w(u, v) + e(v), 0) & \text{if } u \in V_B \end{cases}.$$

Since $e(u)$ is a constant in the minimization and maximization terms, we get

$$e^*_{G,w'}(u) = \begin{cases} \min_{(u,v)\in E} \max(e^*_{G,w'}(v) - w(u,v) - e(u) + e(v), 0) & \text{if } u \in V_A \\ \max_{(u,v)\in E} \max(e^*_{G,w'}(v) - w(u,v) - e(u) + e(v), 0) & \text{if } u \in V_B \end{cases}.$$

By the definition of w' this is equivalent to

$$e^*_{G,w'}(u) = \begin{cases} \min_{(u,v)\in E} \max(e^*_{G,w'}(v) - w'(u,v), 0) & \text{if } u \in V_A \\ \max_{(u,v)\in E} \max(e^*_{G,w'}(v) - w'(u,v), 0) & \text{if } u \in V_B \end{cases}.$$

which is true by Lemma 15. □

5.2 Full Proof of Lemma 14

Algorithm 3: Computing minimal energy based on approximation

Input: A weighted graph (G, w), an upper bound M on the finite minimal energy of (G, w) and a lower bound D on the penalty of (G, w)
Output: The minimal energy of (G, w)

Procedure `MinimalEnergy(`G, w, M, D`)`

```
1  if D ≤ M/(2n) then
2      if M ≤ n then
           // cf. Proposition 5
3          return ValueIteration(G, w, {0, ..., n, ∞})
4      else
5          c ← ⌊M/2⌋
6          e ← Approximate(G, w, M, c)                    // cf. Theorem 2
           // Now solve (G, w') with weights modified by energy e
7          w'(u, v) ← w(u, v) + e(u) − e(v) for every edge (u, v) ∈ G
8          e' ← MinimalEnergy(G, w', c, D)
9          e''(v) ← e(v) + e'(v) for every node v ∈ G
10         return e''
11 else
12     c ← nD
13     e ← Approximate(G, w, M, c)                        // cf. Theorem 2
       // Now solve (G, w') with weights modified by energy e
14     w'(u, v) ← w(u, v) + e(u) − e(v) for every edge (u, v) ∈ G
15     e' ← MinimalEnergy(G, w', c, D)
16     e''(v) ← e(v) + e'(v) for every node v ∈ G
17     return e''
```

Our algorithm is called `MinimalEnergy` and is described in Algorithm 3. We call the algorithm provided by Theorem 2, which computes an approxima-

tion of the minimal energy, `Approximate` and we call the value iteration algorithm provided by Proposition 5, which computes the minimal energy exactly, `ValueIteration`.

We first consider the case $D \geq M/(2n)$. As pointed out in the proof idea, the correctness of `MinimalEnergy` in this case follows from Theorem 2 and Lemma 16. We therefore only argue about the running time. If $M \leq n$, we know that n is an upper bound on the finite minimal energy, and we can use the value iteration algorithm of Proposition 5 with the admissible list $\{0, \ldots, n, \infty\}$, as explained in Section 3. The running time in this case is $O(mn)$. The algorithm `Approximate` runs in time $O(mMn/c)$ for the upper bound M on the finite minimal energy. For $c = \lfloor M/2 \rfloor$ the factor M cancels itself and therefore the running time of `Approximate` is $O(mn)$. We recurse with the upper bound $M' = c = \lfloor M/2 \rfloor$ on the finite minimal energy and the unchanged lower bound D on the penalty. It is still the case that $D \geq M'/(2n)$. Thus, the running time of the procedure `MinimalEnergy` is given by the following recurrence:

$$T(n, m, M) = \begin{cases} O(mn) & \text{if } M \leq n \\ T\left(n, m, \frac{M}{2}\right) + O(mn) & \text{otherwise} \end{cases}.$$

Since the initial value of M is halved with every iteration of the algorithm until $M \leq n$, the algorithm runs for at most $\log M - \log n = \log(M/n)$ many iterations. Every iteration needs time $O(mn)$ and therefore the total running time is $O(mn \cdot \log(M/n))$.

We now consider the case $D < M/(2n)$ in which we perform one step of `Approximate` to reduce M to M' such that $D \geq M'/2n$. We first compute an approximation e of the minimal energy by calling `Approximate` with the approximation error $c = nD$. Then we set $w'(u, v) = w(u, v) + e(u) - e(v)$. We can compute the approximation of the minimal energy in time $O(mM/D)$. After that we can recurse on (G, w') with the new upper bound $M' = c = nD$ on the finite minimal energy to compute $e'(v) = e_{G,w'}(v)$ for every node v. By Lemma 16 the algorithm afterwards correctly returns the minimal energy $e''(v) = e(v) + e'(v)$ for every node v. The new upper bound M' fulfills the following inequality:

$$\frac{M'}{2n} = \frac{nD}{2n} = \frac{D}{2} < D\,.$$

Since the penalty does not change, i.e., $P(G, w) = P(G, w')$ by Lemma 16, our previous running time analysis of the case $D \geq M'/(2n)$ now applies. The remaining time needed to compute the minimal energy of (G, w') therefore is $O(mn \cdot \log(M'/n)) = O(mn \cdot \log D)$. Thus, the total running time in this case is $O(mn \cdot \log D + m \cdot M/D)$. This completes the proof of Lemma 14. □

6 Hardness on Complete Bipartite Graphs

We show in this section that energy games on complete bipartite graphs are polynomial-time equivalent to the general case. This implies that energy games

on graphs of bounded clique-width [11] and strongly ergodic[16] graphs [24] are as hard as the general case.[17] Our result indicates that structural properties of the input graphs might not yield efficiently solvable subclasses. This is in contrast to the fact that parity games (a natural subclass of energy and mean-payoff games) can be solved in polynomial time in these cases [28, 24].

Our main hardness result is for the *decision* problem of energy games which will imply the hardness of the *value* problem as well as of mean-payoff games. The value problem is what we have discussed so far. The *decision problem* of energy games for a graph (G, w) and a node s asks whether the minimal energy $e^*_{G,w}(s)$ is finite. If $e^*_{G,w}(s)$ is finite, we say that Alice *wins at* s; otherwise, we say that Alice loses (or equivalently Bob wins). The decision problem and the value problem of energy games are polynomial-time equivalent [6].[18]

We show that the decision problem on strongly ergodic graphs or graphs of bounded clique-width is just as hard as the general decision problem on arbitrary graphs. For this purpose we will work with a special type of *complete bipartite* graphs which are strongly ergodic and have bounded clique-width (see Definition 20). We note the following fact, proved in Section 6.2.

Lemma 17. *Every complete bipartite graph has clique-width two and is strongly ergodic.*

Our main result is a polynomial-time reduction from the decision problem on arbitrary graphs to the decision problem on complete bipartite graphs.

Theorem 18. *The decision problem of energy games on complete bipartite graphs is polynomial-time equivalent to the decision problem of energy games on general graphs.*

This shows that if we can solve the decision problem of energy games on very special graphs that have clique-width two and are strongly ergodic, then we can solve this problem on general graphs too.

The relationship in Theorem 18 also carries over to the value problem and to mean-payoff games.

Corollary 19. *The value problem of energy games on complete bipartite graphs is polynomial-time equivalent to the value problem of energy games on general graphs. Moreover, the mean-payoff game problem on complete bipartite graphs is polynomial-time equivalent to the mean-payoff game problem on general graphs.*

[16] There are many notions of ergodicity [24, 5]. Strong ergodicity is the strongest one as it implies other ergodicity conditions.

[17] We formally define the notion of *clique-width* and the class of *strongly ergodic* graphs in Section 6.2.

[18] The reduction of Bouyer et al. [6] adds nodes and edges such that, after every edge that is taken, Bob has the possibility to return to the starting node s by an edge of weight t, for a finite $t \geq 0$. In this way we have $e^*_{G,w}(s) \leq t$ if and only if Alice wins at s. It is now possible to find $e^*_{G,w}(s)$ by binary search because the maximum finite energy is limited to nW.

The first statement of the above corollary follows from the fact that the value problem of energy games on general graphs can be reduced to the decision problem [6], and the decision problem on complete bipartite graph is a special case of the value problem on complete bipartite graphs (because solving the value problem also answers the decision problem). For the second statement observe that the decision problem of energy games and the decision problem of mean-payoff games are exactly the same problem [6] because the minimal energy at a node v is finite if and only if the mean-payoff value at v is non-negative. Therefore the statement follows from the fact that the value problem of mean-payoff games can be reduced to the decision problem of mean-payoff games [18], and the decision problem of mean-payoff games on complete bipartite graphs is a special case of the value problem of mean-payoff games on complete bipartite graphs.

The rest of this section is devoted to proving Theorem 18. We first give a proof idea in Section 6.1. In Section 6.2, we formally define the notion of complete bipartite graphs in the context of energy games and prove Lemma 17. In Section 6.3 and 6.4, we show two parts of our reduction. In the first part (Section 6.3), we reduce from the general decision problem of energy games to the problem where it is *promised* that one player wins everywhere, i.e., either the minimal energy function is finite at all nodes (Alice wins) or infinite at all nodes (Bob wins). Note that an input graph of this promised problem is still a general graph. In the second part (Section 6.4), we reduce from this win-everywhere problem on general graphs to the same problem on *complete bipartite* graphs.

6.1 Proof Ideas of Theorem 18

The main idea of proving Theorem 18 is to add "useless" edges to the input graph to make the graph complete bipartite while the answer to the energy game problem remains the same. To illustrate this point, consider any input graph (G, w) and a node s. For each node u belonging to Alice, we add an edge (u, v) with weight $-\infty$, for all nodes v belonging to Bob. Let (G', w') be the new graph.[19]

Observe that if Alice wins in (G, w), i.e. $e^*_{G,w}(s) < \infty$, then she can still play the same strategy in (G', w') so that she wins in (G', w'), i.e. $e^*_{G',w'}(s) < \infty$. On the other hand, if Alice loses in (G, w), i.e. $e^*_{G,w}(s) = \infty$, then the only way she can win in (G', w') is to use some edges that are not in (G, w). These edges, however, have weight $-\infty$. So, the minimum energy that Alice requires remains ∞ even when she use the new edges. Thus, Alice also loses in (G', w'), i.e. $e^*_{G',w'}(s) = \infty$.

The actual proof of Theorem 18 is based on this idea but needs a bit more work. This is because we cannot add an edge of weight $-\infty$. We instead add

[19] Readers that are familiar with parity games might wonder why the same idea does not work for parity games. In parity games every node has a priority. This corresponds to the case where all outgoing edges of a node have the same weight. Under this restriction we are not allowed to add edges of weight $-\infty$ wherever we want to.

an edge of weight $-X$, for large enough X. But this does not solve the whole problem since, when $e^*_{G,w}(s) = \infty$, Alice can use this edge to "escape" to some node v such that $e^*_{G,w}(s) < \infty$. This will make $e^*_{G',w'}(s) < \infty$. To get around this, we first reduce the problem on (G, W) to another graph (G'', w'') where Alice either wins everywhere or loses everywhere. This makes the escaping impossible. We do this in Section 6.3. After we have reduced to the case where one player wins everywhere, we can add edges as above. We also have to add edges from Bob's nodes to Alice's nodes. We assign to these edges a large *positive* weight.

6.2 Properties of Complete Bipartite Graphs

In the following we define what we mean by the class of complete bipartite graphs and show that these graphs are strongly ergodic and have clique-width two. Later, we will show that we can reduce energy games on arbitrary graphs to energy games on complete bipartite graphs.

Definition 20. *A* complete bipartite graph *is a graph $G = (V, E)$ fulfilling the following two conditions:*

- (bipartite) *There is no edge (u, v) from a node $u \in V_A$ of Alice to a node $v \in V_A$ of Alice and there is no edge (u, v) from a node $u \in V_B$ of Bob to a node $v \in V_B$ of Bob.*
- (complete) *For every node $u \in V_A$ of Alice and every node $v \in V_B$ there is an edge $(u, v) \in E$ and for every node $u \in V_B$ of Bob and every node $v \in V_A$ of Bob there is an edge (u, v).*

Note that the number of nodes of Alice and Bob is not required to be equal to fit this definition. We claim that every complete bipartite graph has clique-width two and is strongly ergodic.

The notion of clique-width was introduced by Courcelle and Olariu [11]. We state the definition of clique-width using different notation.

Definition 21. *The* clique-width *of a graph is the minimum number of labels needed to construct G by means of the following four operations.*

1. *Creation of a new node with label i*
2. *Disjoint union of two labeled graphs*
3. *Adding an edge (u, v) for every vertex u with label i and every vertex v with label j*
4. *Renaming label i to label j*

It is easy to see that every complete bipartite graph has clique-width 2. Note that 2 is the smallest clique-width possible for a graph with more than one node. Furthermore, every graph that has bounded tree-width also has bounded clique-width [11]. The concept of tree-width is applied to directed graphs by viewing every edge as an undirected edge. Remember that parity games, which can be reduced to energy games in polynomial-time, can be solved in polynomial time on graphs of bounded clique-width [28].

We now show that every complete bipartite graph is strongly ergodic.

Definition 22 (Ergodicity). [20] *An* ergodic partition *is a pair (S_A, S_B) of sets of nodes such that S_A and S_B are a partition of the nodes satisfying the following conditions:*

1. *For every node u in $S_A \cap V_A$ there is a node $v \in S_A$ such that $(u, v) \in E$; i.e., Alice can always keep the car inside S_A if she wants to.*
2. *There is no edge (u, v) such that $u \in S_A \cap V_B$ and $v \in S_B$; i.e., Bob cannot move the car out of S_A.*
3. *For every node u in $S_B \cap V_B$ there is a node $v \in S_B$ such that $(u, v) \in E$; i.e., Bob can always keep the car inside S_B if he wants to.*
4. *There is no edge (u, v) such that $u \in S_B \cap V_A$ and $v \in S_A$; i.e., Alice cannot move the car out of S_B.*

A graph is ergodic *if it has no non-trivial ergodic partition (a partition (S_A, S_B) is trivial if $S_A = \emptyset$ or $S_B = \emptyset$). A graph is* strongly ergodic *if every induced subgraph such that every node has out-degree at least 1 is ergodic.*

Lemma 23. *Every complete bipartite graph is strongly ergodic.*

Proof. Note that every induced subgraph of a complete bipartite graph is also a complete bipartite graph. Therefore it is sufficient to show that every complete bipartite graph is ergodic.

Suppose that there is a complete bipartite graph G that is not ergodic. Then G has a non-trivial ergodic partition (S_A, S_B). We consider three cases where each one leads to a contradiction:

- S_A contains a node u of Bob, and S_B contains a node v of Alice: Since we have a complete bipartite graph there is an edge (u, v). This means that Bob has an edge leaving S_A which contradicts the second condition in Definition 22.
- S_B contains no node of Alice (S_A might or might not contain a node of Bob): Then S_B only contains nodes of Bob. Since the graph is bipartite all nodes of S_B only have edges that leave S_B. Since S_B is nonempty, there is a node of Bob in S_B that has no edge that stays in S_B which contradicts the third condition in Definition 22.
- S_A contains no node of Bob (S_B might or might not contain a node of Alice): symmetric to previous case.

Since $S_A \neq \emptyset$ and $S_B \neq \emptyset$ we have considered all cases. □

We remark that every graph that is *strongly ergodic* is also *structurally ergodic* in the sense of Boros et al. [5]. Thus, complete bipartite graphs are also structurally ergodic.

6.3 Reduction to Graphs Where One Player Wins Everywhere

In the following, we give the first reduction. We show that energy games on arbitrary weighted graphs can—in polynomial time—be reduced to energy games on weighted graphs in which one player wins at every node.

[20] We use Lebedev's definitions [24].

Lemma 24. *For energy games, the following variants of the decision problem are polynomial-time equivalent:*

- *Decision problem on arbitrary weighted graphs.*
- *Decision problem on weighted graphs in which one player wins everywhere.*

Clearly, graphs in which one player wins everywhere are included in the class of all graphs. The reduction from arbitrary graphs to graphs in which one player wins everywhere goes as follows. We are given a graph (G, w) and want to solve the decision problem, i.e., we want to figure out which player wins at a node s in (G, w). We construct a graph (G', w') as follows. All nodes of G also appear in G' and belong to the same player as in G. We replace every edge (x, y) of G (see Fig. 5) by the following construction: We add a node u of Alice and node v of Bob and add the edges (x, u), (u, v), (v, y), (u, s), and (v, s) with the weights $w'(x, u) = w(x, y)$, $w'(u, v) = w'(v, y) = 0$, $w'(u, s) = -nW$, and $w'(v, s) = nW$.

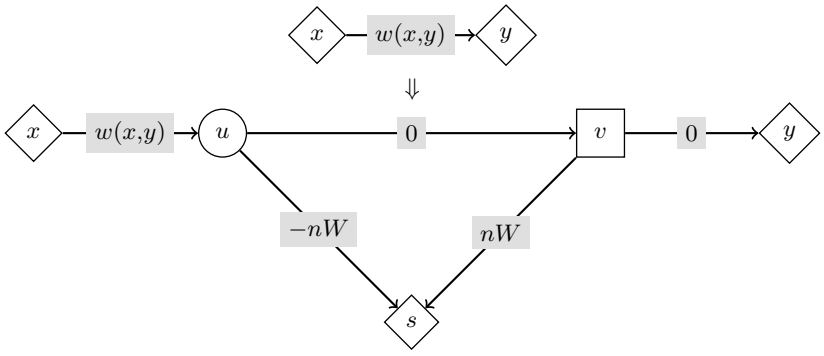

Fig. 5. This picture shows the reduction from the general decision problem to the decision problem in which one of the players wins everywhere. The round nodes belong to Alice and the rectangular nodes belong to Bob. The diamond-shaped nodes are unspecified and could belong to any of the two players.

Lemma 25. *Alice wins at s in (G, w) if and only if Alice wins at s in (G', w').*

Proof. We first prove the following claim: If Alice wins at s in (G, w), then Alice also wins at s in (G', w'). Alice simply has to play the winning strategy σ^* for s in (G, w).[21] If Bob never plays a new edge that goes back to s, his strategy was also available in (G, w) and then Alice wins because σ^* is a winning strategy in G. As soon as Bob plays one of the new edges, a cycle is formed. The cycle C consists of a simple path P from s to some node v and then an edge from v to s. Since the path P in (G', w') does not contain an edge going to s, it corresponds

[21] To be precise: Alice has to play σ^* for nodes already present in (G, w) and for the other nodes the edge that does *not* go back to s has to be chosen.

to some path in (G, w) of the same weight. As a simple path in (G, w) contains at most $n-1$ edges each of weight at least $-W$, the weight of P is at least $-(n-1)W$. Since the edge from x to s has weight nW, the cycle C has positive weight. Therefore σ^* is also a winning strategy in (G', w').

A symmetric argument can be used to prove the following claim: If Bob wins at s in (G, w), then Bob also wins at s in (G', w'). Now the lemma follows from determinacy: Alice does not win if and only if Bob wins. □

Lemma 26. *One of the players wins everywhere in* (G', w').

Proof. We show that the player that wins at s in (G, w) is the one that wins everywhere in G'. We assume that Alice wins at s in (G, w). (For Bob the argument is symmetric.) By Lemma 25 it follows that Alice wins at s in (G', w') by playing some strategy σ. We define a strategy σ' for every node v of Alice as follows: If the edge (v, s) does not exist, we set $\sigma'(v) = \sigma(v)$. If the edge (v, s) does exist we distinguish two cases. If Alice wins at v in (G', w') by playing according to σ, then $\sigma'(v) = \sigma(v)$. Otherwise, Alice takes the new edge that goes to s, i.e., $\sigma(v) = s$. In other words, σ' is defined as follows for every node v of Alice:

$$\sigma'(v) = \begin{cases} s & \text{if edge } (v, s) \text{ exists in } G' \text{ and Alice loses at } v \text{ in } (G', w') \text{ by playing } \sigma \\ \sigma(v) & \text{otherwise} \end{cases}$$

We now show that with the strategy σ' Alice wins against any strategy τ of Bob. Let P be the (unique) infinite path in $(G'(\sigma, \tau), w')$ starting at s.[22] Since σ is a winning strategy of Alice starting from s in (G', w'), Alice wins for every node on P in (G', w') by playing according to σ. By the above definition of σ' we have $\sigma'(v) = \sigma(v)$ for every node v on P. This means that the infinite path in $(G'(\sigma', \tau), w')$ starting at s is exactly P and contains a non-negative cycle.

We now show that in fact for every node u, the infinite path P' in $(G'(\sigma', \tau), w')$ starting at u contains a non-negative cycle. If P' contains s, then P' ends in P. As argued above, P contains a non-negative cycle and therefore also P' contains a non-negative cycle. Consider now the case that P' does not contain s which implies that $\sigma'(v) = \sigma(v)$ for every node v of Alice on P' (because otherwise P' would contain s). Therefore P' is equal to the infinite path in $(G'(\sigma, \tau), w')$ starting at u. By the way we constructed G', P' must contain at least one node v of Alice that has an edge (v, s) to s. Since $\sigma'(v) = \sigma(v) \neq s$ it follows by the way we defined σ' that Alice wins at v in (G', w') by playing according to σ. Therefore P' contains a non-negative cycle as desired. Since τ was an arbitrary strategy of Bob, we know that Alice wins everywhere in (G', w') with the strategy σ'. □

[22] Because of its special structure such a path P is also known as a "lasso" in the literature.

6.4 Reduction to Complete Bipartite Graphs

We now give our second reduction. We show how to reduce the decision problem on graphs in which one player wins everywhere to the decision problem on complete bipartite graphs, as in the following lemma.

Lemma 27. *For energy games, the following variants of the decision problem are polynomial-time equivalent.*

(1) Decision problem on graphs in which one player wins everywhere.
(2) Decision problem on weighted complete bipartite graphs.

Note that the reduction from (2) to (1) is trivial. This is because complete bipartite graphs are strongly ergodic, and in strongly ergodic graphs one player wins everywhere (because otherwise the sets of winning nodes of Alice and Bob, respectively, would immediately give a non-trivial ergodic partition).

The rest of this subsection is devoted to showing the reduction from (1) to (2). This reduction has two parts. We first reduce from (1) to bipartite graphs, which can be done very easily, and from there we reduce to complete bipartite graphs.

Part 1: Reduction to Bipartite Graphs We are given a graph (G, w) in which one of the players wins everywhere. We want make the graph bipartite, i.e., there should neither be an edge (u, v) such that $u \in V_A$ and $v \in V_A$ nor should there be and edge (u, v) such that $u \in V_B$ and $v \in V_B$. We modify (G, w) as follows:

- We replace every edge $(u, v) \in E$ such that $u, v \in V_A$ by two edges (u, u') and (u', v) where u' is a new node of Bob and the weights of the new edges are $w_0(u, u') = w(u, v)$ and $w_0(u', v) = 0$.
- We replace every edge $(u, v) \in E$ such that $u, v \in V_B$ by two edges (u, u') and (u', v) where u' is a new node of Alice and the weights of the new edges are $w_0(u, u') = w(u, v)$ and $w_0(u', v) = 0$.

We call the resulting graph (G_0, w_0). Observe that $e^*_{G,w}(v) = e^*_{G_0,w_0}(v)$ for every node v of G. Therefore the player that wins everywhere in (G, w) also wins everywhere in (G_0, w_0). The same reduction has recently been considered for mean-payoff games by Boros et al. [4].

Part 2: Reduction to Complete Bipartite Graphs We are given a bipartite graph (G, w) in which one player wins everywhere. The reduction to complete bipartite graphs has two steps:

1. Modification of (G, w): For every pair (u, v) of nodes such that $u \in V_A$ and $v \in V_B$, if the edge (u, v) is not contained in G, we add it with weight $w_1(u, v) = -nW$. We call the resulting graph (G_1, w_1).

2. Modification of (G_1, w_1): For every pair (u, v) of nodes such that $u \in V_B$ and $v \in V_A$, if the edge (u, v) is not contained in G, we add it with weight $w_2(u, v) = n^2W$. We call the resulting graph (G_2, w_2).

Clearly (G_2, w_2) is a complete bipartite graph.

Lemma 28. *In (G, w) and (G_2, w_2) the same player wins everywhere.*

Proof. The following claims follow easily:

- If Alice wins everywhere in (G, w), then Alice also wins everywhere in (G_1, w_1). (Alice simply has to play the same strategy as in (G, w), Bob does not have more strategies than in (G, w).)
- If Bob wins everywhere in (G_1, w_1), then Bob also wins everywhere in (G_2, w_2). (Bob simply has to play the same strategy as in (G_1, w_1), Alice does not have more strategies than in (G_1, w_1).)

Now we show the following: If Bob wins everywhere in (G, w), then Bob also wins everywhere in (G_1, w_1). Let τ^* be a winning strategy of Bob in (G, w). We argue that τ^* is also a winning strategy of Bob in (G_1, w_1). Let σ_1 be an arbitrary strategy of Alice in (G_1, w_1). Let C be a cycle in $G_1(\sigma_1, \tau^*)$. If all edges of C already occur in (G, w), we know that C is a cycle of negative weight in (G_1, w_1) because τ^* is a winning strategy of Bob in G. If there is an edge in C that did not already occur in (G, w), then this edge has weight $-nW$. Since the largest positive weight in (G_1, w_1) is W, and C consists of at most n edges, we know that C is a cycle of negative weight. Thus, every cycle in $G(\sigma_1, \tau^*)$ has negative weight. Since σ_1 was an arbitrary strategy of Alice in (G_1, w_1), we conclude that τ^* is a winning strategy of Bob in (G_1, w_1) which he can play to win everywhere.

A symmetric argument can be used to prove the following: If Alice wins everywhere in (G_1, w_1), then Alice also wins everywhere in (G_2, w_2). The only difference to before is that the minimal negative edge weight in (G_1, w_1) is $-nW$ which is the reason why we have to set the weight of the new edges of Bob to n^2W. Since either Alice wins everywhere in (G, w) or Bob wins everywhere in (G, w) it follows by our claims that the same player wins everywhere in (G, w) and (G_2, w_2). □

7 Conclusion

In this paper we answer the question whether the energy game problem can be solved efficiently under certain restrictions. We give both negative and positive answers to this question. On the negative side, we show that usual *graph structure restrictions*, namely clique-width and strong ergodicity, do not make the problem easier. This is in contrast to the situation of the parity game problem (a special case of the energy game problem), which can be solved in polynomial time under such restrictions. Thus, our result provides evidence that energy games might really be harder to solve than parity games.

On the positive side, we identify two *weight structure restrictions* that allow us to solve the energy game problem efficiently: fixed-window and large penalty restrictions. We also provide an algorithm for solving the energy game problem with additive error and show how to use this algorithm to solve the energy game problem exactly.

Many problems remain open for solving energy games and related problems. The most fundamental one is, of course, settling the complexity status of these problems. On the one hand, current algorithmic techniques seem to be insufficient to show that these problems can be solved in polynomial time. On the other hand, it is unlikely that these problem are hard for any complexity classes currently known. It is interesting to investigate how weight structures can help in attacking these problems. For example, it might be possible to transform a graph (G, w) to another graph (G', w') whose penalty is large while the solution to the energy game problem remains the same. While this might be true for *any* graph (G, w), we believe that it is already interesting to show this for some natural class of graphs, e.g. bounded tree-width graphs and graphs from the special case of parity games.

References

1. Beffara, E., Vorobyov, S.: Is randomized Gurvich-Karzanov-Khachiyan's algorithm for parity games polynomial? Tech. Rep. 2001-025, Department of Information Technology, Uppsala University (2001)
2. Björklund, H., Vorobyov, S.G.: A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. Discrete Applied Mathematics 155(2), 210–229 (2007)
3. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better quality in synthesis through quantitative objectives. In: CAV. pp. 140–156 (2009)
4. Boros, E., Elbassioni, K., Gurvich, V., Makino, K.: On canonical forms for zero-sum stochastic mean payoff games. Dynamic Games and Applications to appear(to appear), to appear (2013)
5. Boros, E., Elbassioni, K.M., Fouz, M., Gurvich, V., Makino, K., Manthey, B.: Stochastic mean payoff games: Smoothed analysis and approximation schemes. In: ICALP. pp. 147–158 (2011)
6. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: FORMATS. pp. 33–47 (2008)
7. Brim, L., Chaloupka, J., Doyen, L., Gentilini, R., Raskin, J.F.: Faster algorithms for mean-payoff games. Formal Methods in System Design 38(2), 97–118 (2011)
8. Cerný, P., Chatterjee, K., Henzinger, T.A., Radhakrishna, A., Singh, R.: Quantitative synthesis for concurrent programs. In: CAV. pp. 243–259 (2011)
9. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: EMSOFT. pp. 117–133 (2003)
10. Condon, A.: The complexity of stochastic games. Information and Computation 96(2), 203–224 (1992)
11. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete Applied Mathematics 101(1-3), 77–114 (2000)

12. Daskalakis, C., Papadimitriou, C.H.: Continuous local search. In: SODA. pp. 790–804 (2011)
13. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. International Journal of Game Theory 8(2), 109–113 (1979)
14. Friedmann, O.: A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In: IPCO. pp. 192–206 (2011)
15. Friedmann, O., Hansen, T.D., Zwick, U.: A subexponential lower bound for the random facet algorithm for parity games. In: SODA. pp. 202–216 (2011)
16. Friedmann, O., Hansen, T.D., Zwick, U.: Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In: STOC. pp. 283–292 (2011)
17. Gentilini, R.: A note on the approximation of mean-payoff games. In: CILC (2011)
18. Gurvich, V.A., Karzanov, A.V., Khachiyan, L.G.: Cyclic games and an algorithm to find minimax cycle means in directed graphs. USSR Computational Mathematics and Mathematical Physics 28(5), 85–91 (1990)
19. Halman, N.: Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. Algorithmica 49(1), 37–50 (2007)
20. Jurdzinski, M.: Deciding the winner in parity games is in $\mathsf{UP} \cap \mathsf{co\text{-}UP}$. Information Processing Letters 68(3), 119–124 (1998)
21. Kalai, G.: A subexponential randomized simplex algorithm. In: STOC. pp. 475–482 (1992)
22. Kalai, G.: Linear programming, the simplex algorithm and simple polytopes. Mathematical Programming 79(1-3), 217–233 (1997)
23. Karp, R.M.: A characterization of the minimum cycle mean in a digraph. Discrete Mathematics 23(3), 309–311 (1978)
24. Lebedev, V.N.: Effectively solvable classes of cyclical games. Journal of Computer and Systems Sciences International 44(4), 525–530 (2005)
25. Lifshits, Y.M., Pavlov, D.S.: Potential theory for mean payoff games. Journal of Mathematical Sciences 145(3), 4967–4974 (2007)
26. Martin, D.A.: Borel determinacy. Annals of Mathematics 102(2), 363–371 (1975)
27. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica 16(4-5), 498–516 (1996)
28. Obdrzálek, J.: Clique-width and parity games. In: CSL. pp. 54–68 (2007)
29. Pisaruk, N.N.: Mean cost cyclical games. Mathematics of Operations Research 24(4), 817–828 (1999)
30. Roth, A., Balcan, M.F., Kalai, A., Mansour, Y.: On the equilibria of alternating move games. In: SODA. pp. 805–816 (2010)
31. Sharir, M., Welzl, E.: A combinatorial bound for linear programming and related problems. In: STACS. pp. 569–579 (1992)
32. Vorobyov, S.: Cyclic games and linear programming. Discrete Applied Mathematics 156(11), 2195–2231 (2008)
33. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoretical Computer Science 158(1&2), 343–359 (1996)