Energy Games in Multiweighted Automata

Uli Fahrenberg, Line Juhl, Kim Guldstrand Larsen, Jiří Srba

Energy Games in Multiweighted Automata*

Uli Fahrenberg[1], Line Juhl[2], Kim G. Larsen[2], and Jiří Srba[2]**

[1] INRIA/IRISA, Rennes Cedex, France
ulrich.fahrenberg@irisa.fr
[2] Aalborg University, Department of Computer Science, Denmark
{linej,kgl,srba}@cs.aau.dk

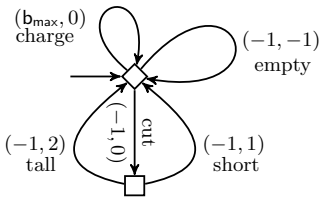
Abstract. Energy games have recently attracted a lot of attention. These are games played on finite weighted automata and concern the existence of infinite runs subject to boundary constraints on the accumulated weight, allowing e.g. only for behaviours where a resource is always available (nonnegative accumulated weight), yet does not exceed a given maximum capacity. We extend energy games to a multiweighted and parameterized setting, allowing us to model systems with multiple quantitative aspects. We present reductions between Petri nets and multiweighted automata and among different types of multiweighted automata and identify new complexity and (un)decidability results for both one and two-player games. We also investigate the tractability of an extension of multiweighted energy games in the setting of timed automata.

1 Introduction

Energy games are two-player games played on finite weighted graphs with the objective of finding an infinite run where the accumulated weight is constrained by a lower and possibly also an upper bound. Such games have attracted considerable attention [3–11, 16] in recent years, as they find natural applications in design and analysis of resource-constrained reactive systems, e.g. embedded or hybrid systems.

We study *multiweighted* energy games, where the weight vectors can have an arbitrary dimension. Let us motivate the study by a small example of an automatic lawn mower with a rechargeable battery and a container for collecting grass. Both the battery and the container have a maximum capacity that cannot be exceeded. We assume that the battery can be recharged and the container can be emptied at nearby servicing stations. The charger is an old-fashioned one, and it charges only for a fixed amount of energy corresponding to going from discharged to fully charged. If the lawn mower starts charging while the battery is not fully discharged, the battery will break. The station for emptying the container removes a unit amount of grass at a time and consumes a unit of battery energy. The container will break if too much grass is stored in it.

* Supported by the VKR Center of Excellence MT-LAB.
** Partially supported by Ministry of Education of Czech Republic, MSM 0021622419.

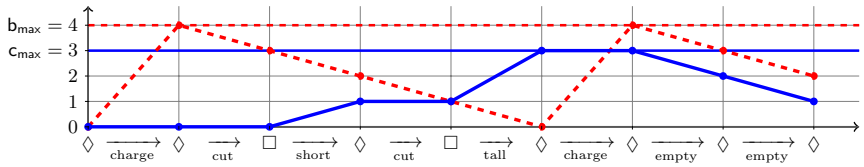


(a) A lawn mower 2-weighted game

$\mathsf{b}_{\max} = 4$ (battery capacity)
$\mathsf{c}_{\max} = 3$ (container capacity)

if *battery* $= 0$ **then** charge
else if *battery* ≥ 2 and *container* ≤ 1 **then** cut
else if *battery* ≥ 1 and *container* ≥ 1 **then** empty

(b) A winning strategy for Player 1



(c) A run of the game (dashed line shows bettery level, solid line container content)

Fig. 1: A lawn mower example

A weighted game describing the lawn mower behaviour is given in Figure 1a. Each transition has a 2-dimensional vector representing the change to the accumulated battery level in the first coordinate and to the accumulated volume of grass in the container in the second coordinate. The numbers $\mathsf{b}_{\max}$ and $\mathsf{c}_{\max}$ represent the maximum capacity of the battery and the container, respectively. The initial state drawn as a diamond is controlled by Player 1 (the existential player), while the other state drawn as a square is controlled by Player 2 (the universal player). In the initial state, Player 1 has the choice of either charging the battery, emptying the container or cutting the grass. Moving to the lawn costs one unit of battery energy, and then Player 2 (the environment) controls whether the actual mowing, which costs again one energy unit, will fill the container with one or two units of grass, depending on whether the grass was short or tall. A *configuration* of the game consists of the state and the accumulated weight in all coordinates. A *run* is a sequence of transitions between configurations formed by the players of the game and starting from the initial state with zero accumulated weight.

The question we ask now (the problem called *energy games with lower and upper bounds*) is whether Player 1 has a strategy so that in the infinite run of actions the lawn mower performs, starting with empty battery and empty container, both the accumulated battery level as well as the container content stay invariantly above zero and do not exceed the given upper bounds $\mathsf{b}_{\max} = 4$ and $\mathsf{c}_{\max} = 3$. Such a strategy exists and is depicted in Figure 1b. Figure 1c illustrates a finite run of the lawn mower game according to this strategy.

If we lower the volume of the container to $\mathsf{c}_{\max} = 2$, no such strategy exists. Player 1 must take the charge transition as the first step, after which cutting is the only opportunity. Player 2 can now choose to cut the short grass, leading

to battery level 2 and grass volume 1. From here Player 1 can only empty the container, as cutting would allow Player 2 to break the container. After emptying the container, battery level is 1 and no transition (apart from cutting) is possible.

There are several variants of the above energy game problem. If we e.g. assume a modern battery charger which does not break the battery when it is not empty, then we have another variant of the problem called *energy games with weak upper bounds*. The weak upper bound game allows taking transitions that will exceed the upper bounds, but these will never accumulate more energy than the maximum capacity. We may also consider infinite runs that are constrained only by a given lower bound but with no upper bound. Finally, we ask questions regarding *parameterization* of the problems. We want to decide whether there *exists* some battery capacity $\mathsf{b}_{\mathsf{max}}$ and some initial battery level such that Player 1 wins the energy game with lower and upper bound (or some of its variants). In our example one can by a simple reasoning argue that for a container capacity $\mathsf{c}_{\mathsf{max}} = 2$, there is no battery capacity $\mathsf{b}_{\mathsf{max}}$ so that Player 1 can guarantee an infinite behaviour of the lawn mower.

Contributions. We define the variants of multiweighted energy games (Section 2) and present reductions involving these games, leading to new decidability and complexity results. Some reductions are to/from Petri nets (Section 3) while others are between different multiweighted energy games (Section 4). This is followed by a a summary of decidability and complexity results we achieved. In Section 6 we consider a parameterized version of existential one-player games and show that some variants of the problem lead to undecidability while others are decidable in polynomial time. We conclude by presenting an undecidability result for a natural timed extension of the energy games (Section 7).

Related work. The idea of checking whether a given resource stays above zero at all times was first presented by Chakrabarti et al. in [6], treating the subject in relation to interfaces. The lower and (weak) upper bound problems were first formulated in [4] for the case with a single weight. The paper presents several complexity results for the 1-weighted case, both timed and untimed, and has given rise to a number of recent papers on 1-weighted energy games [8–10].

The multiweighted extension has been studied in [5], but only for energy games with *unary* weights, i.e. updates by 1, 0 or -1. A continuation of this work presents a polynomial time algorithm for the 2-weighted case with unary inputs [7]. Contrary to this line of work, we consider *binary* input encoding, hence weight updates are now drawn from the full set of integers. Also in contrast to [5, 7], where only complexity *upper* bounds are given, we give complexity *lower* bounds that in most cases match the upper bounds.

Multiweighted energy games with general integer updates have been considered in [8], where the authors show that the problem of deciding the existence of an initial weight vector such that Player 1 can win the lower bound energy game is solvable in polynomial time. In contrast to this, we show here that the non-parameterized variant of this problem: can Player 1 win with a *given* initial weight vector, is EXPSPACE-hard. We also treat the parameterized setting,

where we show that the existential lower and (weak) upper bound problems with both bounds and initial weight vector parameterized are also decidable in polynomial time—unless the upper bound parameter is used in the transitions of the automaton, in which case the problem becomes undecidable.

2 Multiweighted Automata and Games

We denote by $\mathbb{Z}^k$ the set of integer vectors of dimension $k > 0$ and by $\bar{w}[i]$ the i'th coordinate of a vector $\bar{w} \in \mathbb{Z}^k$. A *k-weighted game* G is a four-tuple $(Q_1, Q_2, q_0, \longrightarrow)$ where Q_1 and Q_2 are finite, disjoint sets of *existential* and *universal* states, respectively, $q_0 \in Q_1 \cup Q_2$ is the initial state and $\longrightarrow \subseteq (Q_1 \cup Q_2) \times \mathbb{Z}^k \times (Q_1 \cup Q_2)$ is a finite weighted transition relation, written as $q \xrightarrow{\bar{w}} q'$ whenever $(q, \bar{w}, q') \in \longrightarrow$. Refer to Figure 1a in the introduction for an example of a k-weighted game with $k = 2$.

We are interested only in infinite runs in multiweighted games, hence for the rest of the paper, we assume that the game G is non-blocking, i.e. for every $q \in Q_1 \cup Q_2$ we have $q \xrightarrow{\bar{w}} q'$ for some $\bar{w} \in \mathbb{Z}^k$ and $q' \in Q_1 \cup Q_2$.

A *weighted run* in a k-weighted game $G = (Q_1, Q_2, q_0, \longrightarrow)$ restricted to a *weak upper bound* $\bar{b} \in (\mathbb{N}_0 \cup \infty)^k$ is an infinite sequence $(q_0, \bar{v}_0), (q_1, \bar{v}_1), (q_2, \bar{v}_2), \ldots$ where $q_0, q_1, \ldots \in Q_1 \cup Q_2$, $\bar{v}_0 = \bar{0} = (0, 0, \ldots, 0)$ and $\bar{v}_1, \bar{v}_2, \ldots \in \mathbb{Z}^k$ such that for all $j \geq 0$ we have $q_j \xrightarrow{\bar{w}_j} q_{j+1}$ and

$$\bar{v}_{j+1}[i] = \min\left\{\bar{v}_j[i] + \bar{w}_j[i], \bar{b}[i]\right\}$$

for all coordinates i. An illustration of a run in a 2-weighted game is given in Figure 1c in the introduction. Intuitively, a weighted run is a sequence of states together with the accumulated weight gathered along the path. Moreover, the accumulated weight is truncated, should it exceed in some coordinate the given maximum weight $\bar{b}$. By $\mathsf{WR}_{\bar{b}}(G)$ we shall denote the set of all weighted runs in G restricted to the maximum accumulated weight $\bar{b}$.

A *strategy* for Player $i \in \{1, 2\}$ in a k-weighted game $G = (Q_1, Q_2, q_0, \longrightarrow)$ (restricted to a weak upper bound $\bar{b}$) is a mapping σ from each finite prefix of a weighted run in $\mathsf{WR}_{\bar{b}}(G)$ of the form $(q_0, \bar{v}_0), \ldots, (q_n, \bar{v}_n)$ with $q_n \in Q_i$ to a configuration $(q_{n+1}, \bar{v}_{n+1})$ such that $(q_0, \bar{v}_0), \ldots, (q_n, \bar{v}_n), (q_{n+1}, \bar{v}_{n+1})$ is a prefix of some weighted run in $\mathsf{WR}_{\bar{b}}(G)$. A weighted run $(q_0, \bar{v}_0), (q_1, \bar{v}_1), \ldots$ *respects* a strategy σ of Player i if $\sigma((q_0, \bar{v}_0), \ldots, (q_n, \bar{v}_n)) = (q_{n+1}, \bar{v}_{n+1})$ for all n such that $q_n \in Q_i$. Figure 1b in the introduction shows a strategy for the 2-weighted game from Figure 1a; note that the run of the game depicted in Figure 1c indeed respects this strategy.

We shall consider three decision problems related to energy games on a given k-weighted game $G = (Q_1, Q_2, q_0, \longrightarrow)$. Below we let $\overline{\infty} = (\infty, \infty, \ldots, \infty)$, and we write $\bar{w} \leq \bar{v}$ if $\bar{w}[i] \leq \bar{v}[i]$ for all i, $1 \leq i \leq k$.

Energy Game with Lower bound (GL): Given a game G, is there a strategy σ for Player 1 such that any weighted run $(q_0, \bar{v}_0), (q_1, \bar{v}_1), \ldots \in \mathsf{WR}_{\overline{\infty}}(G)$ respecting σ satisfies $\bar{0} \leq \bar{v}_i$ for all $i \geq 0$?

Hence we ask whether Player 1 has a winning strategy such that during any play the accumulated weight stays above zero in all coordinates.

Energy Game with Lower and Weak upper bound (GLW): Given a game G and a vector of upper bounds $\bar{b} \in \mathbb{N}_0^k$, is there a strategy σ for Player 1 such that any weighted run $(q_0, \bar{v}_0), (q_1, \bar{v}_1), \ldots \in \mathsf{WR}_{\bar{b}}(G)$ respecting σ satisfies $\bar{0} \leq \bar{v}_i$ for all $i \geq 0$?

Hence we ask whether Player 1 has a winning strategy such that during any play the accumulated weight, which is truncated whenever it exceeds the given upper bound, stays in all coordinates above zero.

Energy Game with Lower and Upper bound (GLU): Given a game G and a vector of upper bounds $\bar{b} \in \mathbb{N}_0^k$, is there a strategy σ for Player 1 such that any weighted run $(q_0, \bar{v}_0), (q_1, \bar{v}_1), \ldots \in \mathsf{WR}_{\overline{\infty}}(G)$ respecting σ satisfies $\bar{0} \leq \bar{v}_i \leq \bar{b}$ for all i?

Hence we ask whether Player 1 has a winning strategy such that during any play the accumulated weight stays in all coordinates above zero and below the given upper bound.

The problems GL, GLW and GLU can be specialized in two different ways. Either by giving Player 1 the full control over the game by setting $Q_2 = \emptyset$ or dually by giving the full control to Player 2 by assuming that $Q_1 = \emptyset$. The first problem is called the *existential variant* as we essentially ask whether there *exists* some weighted run with the accumulated weight within the given bounds. The second problem is called the *universal variant* as we now require that *all* weighted runs satisfy the constraints of the energy game. We will denote the respective existential problems by EL, ELW and ELU, and the universal problems by AL, ALW and ALU. These special cases are known as one-player games or simply as multiweighted automata, and we denote such games as only a triple $(Q, q_0, \longrightarrow)$.

In the general formulation of energy games there is no fixed bound on the dimension of the weight vectors, in other words the dimension k is a part of the input. If we want to consider problems of a fixed dimension k, we use the notation GL(k), GLW(k), GLU(k), EL(k) etc.

As the inputs to our decision problems are numbers, it is important to agree on their encoding. We will use the *binary encoding*, unlike some other recent work [5, 7] where unary notation is considered and thus enables to achieve better complexity bounds as the size of their input instance is exponentially larger.

3 Relationship to Petri Nets

We show that infinite run problems on multiweighted automata can be reduced to corresponding problems on Petri nets and vice versa. This allows us to transfer some of the decidability and complexity results from the Petri net theory to our setting. Full proofs and the definition of a Petri net can be found in the appendix.

Lemma 1. *The infinite run Petri net problem is polynomial time reducible to EL. The infinite run Petri net problem for 1-safe nets is polynomial time reducible to ELU and ELW. The problem EL is polynomial time reducible to the infinite run problem of Petri nets.*

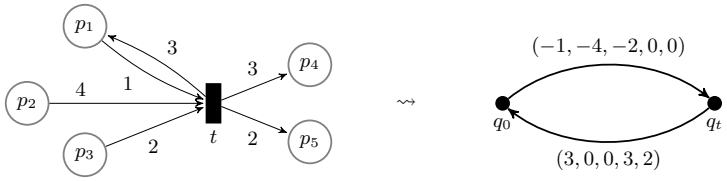


Fig. 2: Translation of a Petri net to a 5-weighted automaton

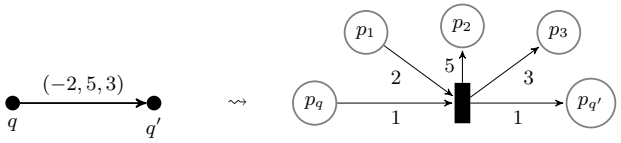


Fig. 3: Translation of a 3-weighted automaton to a Petri net

For the first part of the lemma, a new weight coordinate is introduced for every place in the Petri net and a new state q_t is added for every Petri net transition t, with an additional initial state q_0; see Figure 2. For the second part, we notice that in case the Petri net is 1-safe, one can freely impose an upper bound of $\bar{b} = (1, 1, \ldots, 1)$. For the third part, the states of the automaton are converted to places in the Petri net with an additional place for every weight coordinate and corresponding arcs for updating the markings are added; see Figure 3.

Theorem 1. *The problem EL is EXPSPACE-complete. The problems ELU and ELW are PSPACE-complete.*

Proof. The complexity bounds for EL follow from Lemma 1 above and from the fact that the existence of an infinite run in a Petri net is decidable in EXPSPACE [15] and EXPSPACE-hard (see e.g. [12]). The same problem for 1-safe Petri nets is PSPACE-complete (see again [12]) and by Lemma 1 we get PSPACE-hardness also for ELU and ELW. The containment of the ELU and ELW problems in PSPACE can be shown by noticing that these problems have an infinite run $(q_0, \bar{v}_0), (q_1, \bar{v}_1), \ldots$ if and only if there are two indices $i < j$ such that $(q_i, \bar{v}_i) = (q_j, \bar{v}_j)$. As the size of any configuration $(q, \bar{v})$ appearing on such a run is polynomially bounded by the size of the input (which includes the upper bound vector), we can use a nondeterministic algorithm to guess such a repeated configuration $(q_i, \bar{v}_i)$ and nondeterministically verify whether it forms a loop which is reachable from the initial pair $(q_0, \bar{v}_0)$. This completes the argument for the containment of ELU and ELW in PSPACE. □

4 Reductions among Energy Games

In this section we present reductions among the variants of one and two-player energy games with a particular focus on the size of the weight vectors. Full proofs

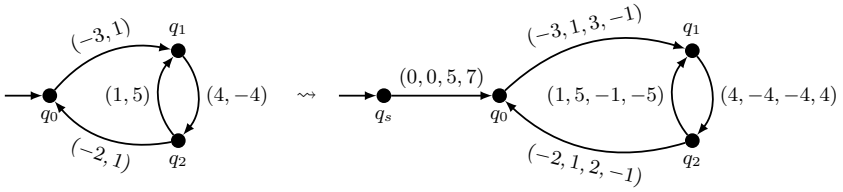


Fig. 4: Example of reduction from GLU with $\bar{b} = (5, 7)$ to GL

are given in the appendix. Our first result can be easily proven by duplicating the coordinates in the weight vector while reversing the sign; see Figure 4.

Theorem 2. *The problem GLU(k) is polynomial time reducible to GL($2k$) and GLW($2k$) for all $k > 0$. The reduction preserves the existential and universal variants of the problems.*

Since we already know that ELU(1) is NP-hard [4], using Theorem 2 with $k = 1$ gives that EL(2) is NP-hard too, which is of course then also the case for EL. Similarly as GLU(1) is known to be EXPTIME-hard [4], we get EXPTIME-hardness also for GL(2) and hence also for GL.

Our next reductions show (perhaps surprisingly) that allowing multiple weights in not that crucial in terms of complexity. The first theorem shows that for upper bound games, it suffices to work with one weight only; Theorem 4 then shows that for the existential variant, two weights are enough.

Theorem 3. *The problem GLU is polynomial time reducible to GLU(1).*

Proof. Let $G = (Q_1, Q_2, q_0, \longrightarrow)$ be a k-weighted game and $\bar{b}$ a given upper bound vector for the GLU problem. We assume that G is encoded in binary and let n denote the size of such encoding. This means that all constants that appear in the description of G are less than 2^n. We will construct a corresponding 1-weighted game $G' = (Q'_1, Q'_2, q_s, \longrightarrow)$ where $Q'_1 = Q_1 \cup \{q_2, q_3, \ldots, q_{k+5} \mid q \in Q_1\} \cup \{q_s\}$ and $Q'_2 = Q_2 \cup \{q_1 \mid q \in Q_2\}$ that simulates G.

Let $\bar{w}$ denote any weight vector present in G. Clearly, $0 \leq \bar{w}[1], \ldots, \bar{w}[k] < 2^n$ due to the encoding of the input. Without loss of generality we can assume that all coordinates of $\bar{b}$ are the same, i.e. that $\bar{b} = (b, \ldots, b)$ for some $0 \leq b < 2^n$.

We need to encode the weights from G using only one weight. We will do so by placing them into the single (large) weight w'. Since $b < 2^n$, at most n bits are needed to represent each weight $\bar{w}[i]$. The weight w' is constructed by appending the weights from G in higher and higher bit positions, with a suitable separation sequence to ensure that weights cannot get 'entangled' should their bounds overflow or underflow. Formally, we introduce the following notation for any integer $\ell \in \mathbb{Z}$ and any i, $1 \leq i \leq k$:

$$\langle \ell \rangle^i = \ell \cdot 2^{(i-1)(n+2)} \ .$$

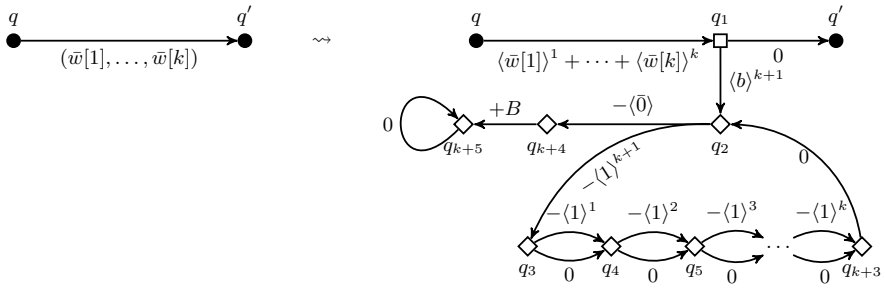


Fig. 5: Simulation of a transition in a k-weighted game by a 1-weighted game

For example, if $n = 4$ then $\langle 6\rangle^2 = 6 \cdot 2^6 =$ (in binary) $= 110 \cdot 1000000 = 110000000$. A weight vector $\bar{w}$ of size k in G is now represented by the number

$$\langle \bar{w}\rangle \stackrel{\text{def}}{=} \langle \bar{w}[1]\rangle^1 + \langle 2^{n+1}\rangle^1 + \langle \bar{w}[2]\rangle^2 + \langle 2^{n+1}\rangle^2 + \ldots + \langle \bar{w}[k]\rangle^k + \langle 2^{n+1}\rangle^k$$

where the weights $\bar{w}[1], \ldots, \bar{w}[k]$ written in binary from the less significant bits to more significant ones are separated by the binary string '10'. For example if again $n = 4$ then the weight vector $\bar{w} = (110, 1, 1011)$ with the weights written in binary is represented by the binary number $\langle \bar{w}\rangle = 10\ 1011\ 10\ 0001\ 10\ 0110$.

The new upper bound B for G' is defined by $B = \langle b\rangle^{k+1} + \langle \bar{b}\rangle$ where apart from the standard encoding of all upper bounds for all coordinates we add one more time the constant b at the most significant bits (we will use these bits for counting in our construction).

Each transition $q \xrightarrow{\bar{w}} q'$ in G is transformed into a number of transitions in G' as depicted in Figure 5 where Player 1 (existential) states are drawn as diamonds and Player 2 (universal) states are drawn as squares. The states drawn as filled circles can be of either type, and their type is preserved in the translation. We also add the initial transition $q_s \xrightarrow{\langle \bar{0}\rangle} q_0$ which inserts the separation strings 10 at the correct positions.

The idea is that the update of the accumulated weight vector $\bar{v}$ in G via adding a vector $\bar{w}$ like in Figure 5 is simulated by adding the numbers $\langle \bar{w}[1]\rangle^1$, $\langle \bar{w}[2]\rangle^2$, $\ldots$, $\langle \bar{w}[k]\rangle^k$ to the accumulated weight in G'. The chosen encoding of k weights into a single weight is crucial to preserve the soundness of the construction as discussed in the following remark.

Remark 1. Given an accumulated weight vector $\bar{v}$ and a weight update vector $\bar{w}$ where $\bar{0} \leq \bar{v}, \bar{w} \leq \bar{b} < (2^n, \ldots, 2^n)$, then adding the numbers $\langle \bar{v}\rangle$ and $\langle \bar{w}[i]\rangle^i$ in $\langle \bar{v}\rangle$ changes at most the bits that are designated for representing the weight coordinate $\bar{w}[i]$ and the separating two bits 10 just before it. This can be easily seen by analyzing the two extreme cases of adding $11\ldots1$ to an accumulated weight coordinate with full capacity and subtracting $11\ldots1$ from an accumulated weight coordinate that represents zero as showed in the following two examples.

$$\begin{array}{r} \ldots 10\ 111 \ldots 111\ 10 \ldots \\ +\ldots 00\ 111 \ldots 111\ 00 \ldots \\ \hline \ldots 11\ 111 \ldots 110\ 10 \ldots \end{array} \qquad \begin{array}{r} \ldots 10\ 000 \ldots 000\ 10 \ldots \\ -\ldots 00\ 111 \ldots 111\ 00 \ldots \\ \hline \ldots 01\ 000 \ldots 001\ 10 \ldots \end{array}$$

Let us now argue about the correctness of this polynomial time construction. Assume that Player 1 has a winning strategy in the game G. As the accumulated weight stays within the bounds during any such play in G, it is clear that the same winning strategy can be performed also in G' using only a single weight. One complication is that each transition in G is split in G' and a new node for Player 2 (q_1 in Figure 5) is inserted. Hence Player 2 could possibly have an extra winning strategy by playing $q_1 \xrightarrow{\langle b \rangle^{k+1}} q_2$, instead of the expected move to q'. However, because the accumulated weight vector $\bar{v}$ satisfies $0 \leq \bar{v}[i] \leq b < 2^n$ for all i, we can see that Player 1 wins in this case, by taking the loop $q_2, q_3, \ldots q_{k+3}, q_2$ exactly b times while choosing the zero or $-\langle 1 \rangle^i$ transitions (for all i) in such a way that the bits representing the weight $\bar{v}[i]$ are all set to 0. What remains in G' as the accumulated weight is then the value $\langle \bar{0} \rangle$ which consists only of the separation symbols. From here Player 1 takes the transition with weight $-\langle \bar{0} \rangle$, setting the accumulated weight to zero, and wins by performing the transition labeled with $+B$ (which is possible only if the accumulated weight is exactly zero) and repeatedly performing in q_{k+5} the self-loop with weight zero.

On the other hand, assume that a play in G causes the accumulated weight in some coordinate i, $1 \leq i \leq k$, to get out of the bounds; we shall argue that Player 2 has a winning strategy in G' in this case. Should this happen during a transition from q to q' in G, then in G', Player 2 will simply move from the intermediate state q_1 to q_2, while the counter value of size b is added to the most significant bits of the accumulated weight via adding the number $\langle b \rangle^{k+1}$. It is clear that it is possible to move from q_2 to q_{k+5} only if the accumulated weight is exactly $\langle \bar{0} \rangle$. In order to achieve this value, the accumulated weight needs to be decreased exactly b times via taking the loop $q_2, q_3, \ldots, q_{k+3}, q_2$. Because of Remark 1 we can see that only the bits relevant to each weight coordinate were changed before entering the loop, so it is impossible to zero all bits corresponding to the coordinate i while preserving the separation bits 10. □

When considering the existential problems, we cannot use only one weight in order to encode the weights with upper and weak upper bounds. Since we do not have the adversary to check for violation of the bounds, we cannot 'destroy' the weights in the overflow checking process, and thus need an extra weight to store the current value of all weights before a check.

Theorem 4. *The problem ELU is polynomial time reducible to ELU*(2).

Proof. The reduction idea is similar to the one in the proof of Theorem 3. The main complication is that Player 2 has no states in control, hence checking the underflow and overflow of weights has to be performed without resorting to an opponent. As the original weight values are destroyed during such a check, we need to employ the second weight for saving them.

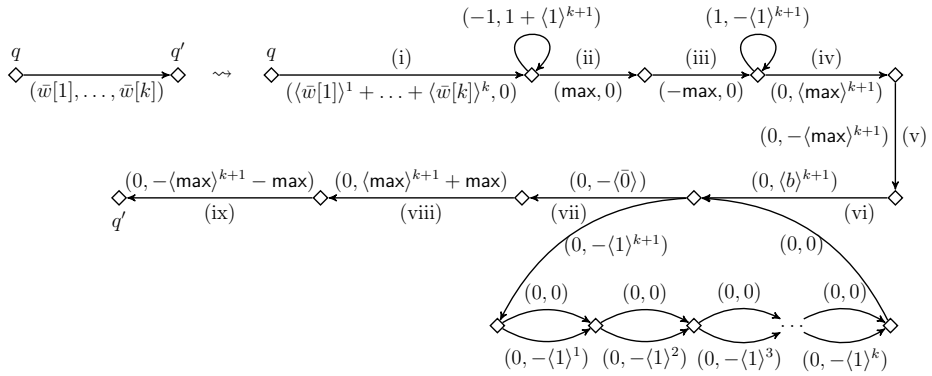


Fig. 6: Simulation of a transition in a k-weighted automaton by a 2-weighted automaton

Let $A = (Q, q_0, \longrightarrow)$ be a k-weighted automaton and $\bar{b}$ the upper bound vector for the ELU problem. We construct a corresponding 2-weighted automaton $A' = (Q', q_s, \longrightarrow)$. Let $\bar{w}$ denote the weight vectors in A and $\bar{v}[1], \bar{v}[2]$ the two weights in A'. As before for an input automaton of size n we may assume that all weights in A have the same upper bound $\bar{b} = (b, b, \ldots, b)$ where $b < 2^n$.

The upper bound $\bar{b}'$ for the ELU(2) problem in A' is given by $\bar{b}' = (\mathsf{max}, \langle \mathsf{max} \rangle^{k+1} + \mathsf{max})$ with $\mathsf{max} = \langle b \rangle^{k+1} + \langle \bar{b} \rangle$. The reason for reserving twice as many bits in the second weight is that we need to save there *two* copies of the first weight. Figure 6 shows how to simulate one transition in A by a number of transitions in A'. From the newly added initial state q_s we also add the transition $q_s \xrightarrow{(\langle \bar{0} \rangle, 0)} q_0$ which inserts the separation strings '10' into the first weight.

We shall now argue that the automaton A' faithfully simulates A. We will examine the effect of the sequence of transitions between q and q' added to the automaton A' (here numbered with (i), (ii), ..., (ix) for convenience) and argue at the same time that the part of the run between q and q' in A' is uniquely determined. By construction, $\bar{v}[2]$ will be zero when entering q, and then the transition (i) adds the encoded weights of the original transition in A to $\bar{v}[1]$. Transition (ii) will add the upper bound to $\bar{v}[1]$, hence before this, we need to take the loop with weight $(-1, 1 + \langle 1 \rangle^{k+1})$ until $\bar{v}[1]$ equals zero, thereby copying twice the value of $\bar{v}[1]$ to $\bar{v}[2]$ (first copy in the less significant bits, second copy in the more significant bits). After the transitions (ii) and (iii), $\bar{v}[1]$ is then again at zero. Now transition (iv) wants to add the upper bound to the most significant bits of $\bar{v}[2]$, hence before this, we need to take the loop with weight $(1, -\langle 1 \rangle^{k+1})$ until the value of the most significant bits in $\bar{v}[2]$ is copied to $\bar{v}[1]$, thereby restoring the original weight in $\bar{v}[1]$.

After the transitions (iv) and (v), we are in a situation where both coordinates in the accumulated weight store the same number, and we can afford to destroy the second copy during the verification phase for bound overflow/underflow performed by transitions (vi), the long loop, and transitions (vii), (viii) and (ix).

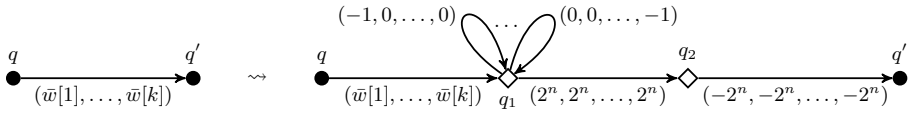


Fig. 7: Simulation of a transition in a LW game by a LU game

This is identical to the construction in the previous proof (except for the extra coordinate $\bar{v}[1]$ which is not updated). Provided that no violation of bounds was detected, we will reach q' with $\bar{v}[1]$ encoding the weight vector of A at q' and $\bar{v}[2]$ equal to zero.

Hence a transition between two states in A can be performed if and only if the sequence of transitions between q and q' in A' can be performed. As the reduction is clearly in polynomial time, this concludes the proof. □

The next theorem finishes our considerations about reductions between different variants of energy games. For the proof, the weak upper bound $\bar{b} = (b, \ldots, b)$ is widened to $(2^n + b, \ldots, 2^n + b)$ (where again n is the size of the binary encoding of the game), and some extra transitions are added to ensure that the weights are truncated at $\bar{b}$; see Figure 7.

Theorem 5. *The problem GLW is polynomial time reducible to GLU, and ELW is polynomial time reducible to ELU.*

Now, in combination with Theorems 3 and 4, we get the following corollary.

Corollary 1. *The problems GLW and ELW are polynomial time reducible to* $GLU(1)$ *and* $ELU(2)$*, respectively.*

5 Summary of Complexity Results

The collection of complexity results and reductions between different types of energy games and automata enables us to draw the conclusions presented in Table 1. Notice that the LU problems are computationally easier than the L problems for an arbitrary number of weights, even though they are harder than the L problems in the 1-weighted case. The configuration space for the LU (and LW) problems is bounded (see Theorem 1), whereas the same a priori does not apply to the L problem.

Observe also that any *universal* problem with k weights can be solved by checking the same problem for each coordinate independently. If the k-weighted problem violates the bounds at some coordinate, so will do the 1-weighted problem projected on this coordinate. On the other hand, if some coordinate in the 1-weighted problem violates the bounds then so will do the k-weighted game, as the same run leading to the violation in one coordinate leads to a violation in the k-weighted game (unless the violation occurs in some other coordinate before that). As AL(1), ALW(1) and ALU(1) are decidable in P [4], this implies polynomial upper bounds also for all the other k-weighted universal problems.

Weights	Type	Existential	Game
One	L	$\in$ P [4]	$\in$ UP $\cap$ coUP [4]
	LW	$\in$ P [4]	$\in$ NP $\cap$ coNP [4]
	LU	NP-hard [4], $\in$ PSPACE [4]	EXPTIME-complete [4]
Fixed (k>1)	L	**NP-hard**, $\in$ k-EXPTIME [5] (Rem. 2)	**EXPTIME-hard**, $\in$ k-EXPTIME [5] (Rem. 3)
	LW	**NP-hard**, $\in$ **PSPACE** (Rem. 4) **PSPACE-complete** for $k \geq 4$	**EXPTIME-complete** (Rem. 5)
	LU	**PSPACE-complete** (Rem. 4)	**EXPTIME-complete** (Rem. 5)
Arbitrary	L	**EXPSPACE-complete** (Thm. 1)	**EXPSPACE-hard** (from EL) decidable [5]
	LW	**PSPACE-complete** (Thm. 1)	**EXPTIME-complete** (Rem. 5)
	LU	**PSPACE-complete** (Thm. 1)	**EXPTIME-complete** (Rem. 5)

Table 1: Complexity bounds; results obtained in this paper are in bold

Remark 2. The problem ELU(1) is NP-hard, and Theorem 2 implies NP-hardness for EL(2). The upper bound follows from the game version of the problem (see also Remark 3).

Remark 3. The lower bound follows from EXPTIME-hardness of GLU(1) and Theorem 2. The upper bound is due to a result in [5] showing ($k-1$)-EXPTIME containment for GL(k) but for games where weight updates are only $+1$, 0, and -1. We can reduce updates with arbitrary weights into this setting by standard techniques (introducing intermediate transitions which repeatedly add or subtract 1) but this causes an exponential blowup in the size of the system. Hence the complexity upper bound increases by one exponent to k-EXPTIME.

Remark 4. The PSPACE upper bound follows from the results for an arbitrary number of weights (Theorem 1). The PSPACE lower bound for ELU(2) is due to the reduction in Theorem 4 and PSPACE-hardness of ELU. By using Theorem 2 we get PSPACE-hardness for ELW(4) because ELU(2) is PSPACE-hard, and we also get NP-hardness of ELW(2) as ELU(1) is NP-hard.

Remark 5. The upper bound for GLU follows from Theorem 3 and the EXPTIME upper bound for GLU(1); the upper bound for GLW follows additionally from Theorem 5. The lower bound for GLU is obvious and for GLW it is by Theorem 2 and the EXPTIME-hardness result for GLU(1).

6 Parameterized Existential Problems

In this section we shall focus in more detail on the existential one-player energy games. So far we have studied decision problems where both the initial weight vector and the upper bound were given as a part of the input. We will now

consider parameterized versions of the problems where, given a weighted automaton, we ask whether there is some initial weight vector $\bar{v}_0$ (and some upper bound $\bar{b}$ in case of ELU and ELW) such that the automaton has a run where the accumulated weight satisfies the constraints imposed by the respective variant of the problem.

Recent work by Chatterjee et al. [8] proves that the parameterized version of the EL problem, asking if there is an initial weight vector such that the accumulated weight of some run in the automaton stays (component-wise) above zero, is decidable in polynomial time. Perhaps surprisingly, this result contrasts with our EXPSPACE-hardness result for the EL problem where the initial weight vector is fixed. An interesting fact, using Lemma 1, is that by the result of [8], it is also decidable in polynomial time whether there is an initial marking such that a given Petri net has an infinite run.

The situation can be, however, different when considering the problems ELU and ELW. Depending on whether the parameterized upper bound $\bar{b}$ is allowed to appear as a weight in transitions of the given weighted automaton (see Section 1 for an example where the upper bound appears as a weight), we shall show below that the problem is either decidable in polynomial time or undecidable.

We present first the positive result. Its proof is based on a polynomial time algorithm for zero-weight cycle detection in multiweighted automata by Kosaraju and Sullivan [13], and we acknowledge [8] where we found a pointer to this result, which is mentioned there in connection with the parameterized EL problem.

Theorem 6. *The parameterized ELU and ELW problems where the upper bound parameter does not appear as a weight in the underlying weighted automaton are decidable in polynomial time.*

However, if the uppper bound can appear as a weight, we get undecidability.

Theorem 7. *The parameterized ELU(2) and ELW(4) problems where the upper bound parameter can appear as a weight in the underlying weighted automaton are undecidable.*

Proof. We provide a reduction from the undecidable halting problem of Minsky machines [14] to ELU(3). Let $1 : \mathsf{inst}_1;\ 2 : \mathsf{inst}_2;\ \ldots, n : \mathsf{inst}_n$ be a Minsky machine over the nonnegative counters c_1 and c_2. We construct a 3-weighted automaton $(Q, q_0, \longrightarrow)$ where $Q = \{q_i, q_i' \mid 0 \leq i \leq n\}$ and where the initial weight vector $\bar{v}_0$ and the upper bound $\bar{b}$ are parameterized. The intuition is that the first and second coordinates will record the accumulated values of counters c_1 and c_2, respectively, and the third coordinate will be used for counting the number of steps the machine performs. The transitions are of four types:

1. $q_0 \xrightarrow{+\bar{b}} q_0' \xrightarrow{-\bar{b}} q_1$
2. For each instruction i: c_j `:=` c_j `+ 1; goto` k, we add the transitions
   - $q_i \xrightarrow{(+1,0,+1)} q_k$ if $j = 1$, and $q_i \xrightarrow{(0,+1,+1)} q_k$ if $j = 2$.
3. For each instruction i: `if` c_j `= 0 then goto` k `else (`c_j `:=` c_j `- 1; goto` ℓ`)`, we add the transitions

- $q_i \xrightarrow{(+\bar{b}[1],0,0)} q_i' \xrightarrow{(-\bar{b}[1],0,+1)} q_k$ and $q_i \xrightarrow{(-1,0,+1)} q_\ell$ if $j = 1$, and
- $q_i \xrightarrow{(0,+\bar{b}[2],0)} q_i' \xrightarrow{(0,-\bar{b}[2],+1)} q_k$ and $q_i \xrightarrow{(0,-1,+1)} q_\ell$ if $j = 2$.

4. Finally, we add the loop $q_n \xrightarrow{(0,0,0)} q_n$.

It is straightforward to argue that the constructed 3-weighted automaton has an infinite run iff the Minsky machine has an infinite computation. From Theorem 4 we get that ELU(3) is reducible to ELU(2), hence the parameterized existential problem is undecidable for vectors of dimension two. By Theorem 2 we can reduce ELU(2) to ELW(4), which implies the undecidability of the problem also for weak upper bound and weight vectors of size at least four. □

The parameterized problems ELU(1) and ELW(k) for $1 \leq k \leq 3$ where the upper bound parameter can appear in the automata are open.

7 Extension to Timed Automata

It is natural to ask for extensions of the results presented in this article to multi-weighted *timed* automata and games [1, 2]. For the case with one weight and one clock only, such extensions have been discussed in [3, 4]. In [4] it has been shown that the GLU(1) problem is already undecidable for one-clock multiweighted timed automata. By an adaptation of the technique introduced in [4], we can prove that the existential problem ELU with two weights and one clock is also undecidable. As the reductions from Theorem 2 apply also to timed automata, we altogether get the following undecidability results.

Theorem 8. *The problems ELU*(2)*, EL*(4) *and ELW*(4)*, and GLU*(1)*, GL*(2) *and GLW*(2) *are undecidable for one-clock multiweighted timed automata.*

8 Conclusion and Future Work

We have presented an extension of different types of energy games to a setting with multiple weights and established a comprehensive account of the complexity of these problems. To derive our results, we have demonstrated a close connection of these problems with infinite run problems in Petri nets, together with a number of reductions between different variants of multiweighted energy games. We have also studied a parameterized version of these problems and shown that depending on the precise statement of the problem, it is either solvable in polynomial time or undecidable. Finally, we have demonstrated that for the timed automata extension of energy games, the lower and upper bound existential problem is undecidable already for one clock and two weights.

There are two main problems left open. The first one deals with settling the complexity of the one-weight lower and upper bound existential problem, which is only known to be between NP and PSPACE. This is closely related to the lower bound and weak upper bound problems with a fixed number of weights. The second problem deals with the complexity of energy games with lower bound only, as the present upper complexity bound depends on the number of weights and does not have a matching lower bound.

References

1. Alur, R., Torre, S.L., Pappas, G.J.: Optimal paths in weighted timed automata. Theoretical Computer Science 318(3), 297–322 (2004)
2. Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In: Benedetto, M.D.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC. Lecture Notes in Computer Science, vol. 2034, pp. 147–161. Springer (2001)
3. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N.: Timed automata with observers under energy constraints. In: Johansson, K.H., Yi, W. (eds.) HSCC. pp. 61–70. ACM ACM (2010)
4. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) Proceedings of the 6th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'08). LNCS, vol. 5215, pp. 33–47. Springer, Saint-Malo, France (Sep 2008)
5. Brázdil, T., Jančar, P., Kučera, A.: Reachability games on extended vector addition systems with states. In: Proceedings of 37th International Colloquium on Automata, Languages and Programming (ICALP'10). LNCS, vol. 6199, pp. 478–489. Springer (2010)
6. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Embedded Software, Third International Conference, EMSOFT 2003, Philadelphia, PA, USA, October 13-15, 2003, Proceedings. LNCS, vol. 2855, pp. 117–133. Springer (2003)
7. Chaloupka, J.: Z-reachability problem for games on 2-dimensional vector addition systems with states is in P. In: Proceedings of the 4th International Workshop on Reachability Problems (RP'10). LNCS, vol. 6227, pp. 104–119. Springer (2010)
8. Chatterjee, K., Doyen, L., Henzinger, T., Raskin, J.F.: Generalized mean-payoff and energy games. In: Proceedings of FSTTCS'10. LIPIcs, vol. 8, pp. 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
9. Chatterjee, K., Doyen, L.: Energy parity games. In: Abramsky, S., Gavoille, C., Kirchner, C., auf der Heide, F.M., Spirakis, P.G. (eds.) ICALP (2). Lecture Notes in Computer Science, vol. 6199, pp. 599–610. Springer (2010)
10. Degorre, A., Doyen, L., Gentilini, R., Raskin, J.F., Torunczyk, S.: Energy and mean-payoff games with imperfect information. In: Dawar, A., Veith, H. (eds.) CSL. Lecture Notes in Computer Science, vol. 6247, pp. 260–274. Springer (2010)
11. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. International Journal of Game Theory 8(2), 109–113 (1979)
12. Esparza, J.: Decidability and complexity of Petri net problems — An introduction. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, LNCS, vol. 1491, pp. 374–428. Springer Berlin / Heidelberg (1998)
13. Kosaraju, S., Sullivan, G.: Detecting cycles in dynamic graphs in polynomial time. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC'88). pp. 398–406. ACM (1988)
14. Minsky, M.: Computation: Finite and Infinite Machines. Prentice-Hall (1967)
15. Yen, H.: A unified approach for deciding the existence of certain Petri net paths. Information and Computation 96(1), 119–137 (1992)
16. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theorertical Computer Science 158(1&2), 343–359 (1996)

Appendix

Remarks about Energy Games. We may without loss of generality assume an initial weight vector $\bar{w}_0$ different from $\bar{0}$. This is evident by adding a new fresh start state with one transition labeled with $\bar{w}_0$ pointing to the original start state. In addition we may assume that in any given upper bound or weak upper bound vector $\bar{b}$ we have $\bar{b}[1] = \bar{b}[2] = \cdots = \bar{b}[k]$. This can be achieved by scaling every i'th coordinate of all weight vectors on transitions with $\frac{\bar{b}[1]\cdot\ldots\cdot\bar{b}[k]}{\bar{b}[i]}$ in order to obtain equality on the coordinates of $\bar{b}$. Such a scaling implies only polynomial increase in the size (in binary encoding) of the upper bound constants.

Definition of Petri nets. We define the Petri net model with weighted arcs (that allow to consume more than one token from a given place).

A *Petri net* is a triple $N = (P, T, W)$ where P is a finite set of places, T is a finite set of transitions, and $W : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ is a function assigning a weight to each arc in the net. A *marking* on N is a function $M : P \to \mathbb{N}_0$ denoting the number of tokens present in the places. A *marked Petri net* is a pair (N, M_0) where N is a Petri net and M_0 is an initial marking on N.

A transition $t \in T$ is *enabled* in a marking M if $M(p) \geq W(p,t)$ for all $p \in P$. An enabled transition may *fire*. When a transition t fires, it produces a new marking M' obtained as $M'(p) = M(p) - W(p,t) + W(t,p)$ for all places $p \in P$. Then we write $M \xrightarrow{t} M'$. A marking M is *reachable* in N if $M_0 \longrightarrow^* M$ where $\longrightarrow = \bigcup_{t \in T} \xrightarrow{t}$. A marked Petri net is called *1-safe* if for any reachable marking M the number of tokens in any place is at most one, i.e. $M(p) \leq 1$ for all $p \in P$.

We say that a marked net (N, M_0) has an infinite run if there is a sequence of markings $M_1, M_2, \ldots$ and transitions $t_1, t_2, \ldots$ such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_3} \ldots$ The *infinite run problem* for Petri nets (see e.g. [12]) is to decide whether a given Petri net has an infinite run.

Proof (of Lemma 1). We first prove the first part of the lemma. Given a Petri net $N = (P, T, W)$ where $P = \{p_1, \ldots, p_k\}$ we construct a k-weighted automaton $A = (Q, q_0, \longrightarrow)$ such that $Q = \{q_0\} \cup \{q_t \mid t \in T\}$. Now for every $t \in T$ we add to A two transitions $q_0 \xrightarrow{\bar{w}_t^-} q_t$ and $q_t \xrightarrow{\bar{w}_t^+} q_0$ where $\bar{w}_t^-[i] = -W(p_i, t)$ and $\bar{w}_t^+[i] = W(t, p_i)$ for all i, $1 \leq i \leq k$. Consult Figure 2 for an example. The initial weight vector then corresponds to the initial marking of the net in the expected way.

It follows from the construction that each transition firing can be simulated by two transitions in the constructed weighted automaton and vice versa. Observe that the reachable Petri net markings are represented as accumulated weight vectors in the automaton and hence are nonnegative in all coordinates. It is easy to verify that the net has an infinite run if and only if the EL problem has a solution. The reduction clearly runs in polynomial time.

For the second part, observe that if the net is 1-safe then by taking the upper bound $\bar{b} = (1, 1, \ldots, 1)$ we have a reduction from the infinite run problem for 1-safe nets to ELU and ELW.

The reduction from k-weighted automata to Petri nets works in a similar way. Given a k-weighted automaton $A = (Q, q_0, \longrightarrow)$ we construct a Petri net $N = (P, T, W)$ where $P = \{p_1, \ldots, p_k\} \cup \{p_q \mid q \in Q\}$ and $T = \{t_{(q,\bar{w},q')} \mid q \xrightarrow{\bar{w}} q'\}$. For each $t_{(q,\bar{w},q')}$ we set $W(p_q, t_{(q,\bar{w},q')}) = 1$, $W(t_{(q,\bar{w},q')}, p_{q'}) = 1$ and for all i, $1 \leq i \leq k$, $W(p_i, t_{(q,\bar{w},q')}) = -\bar{w}[i]$ if $\bar{w}[i] < 0$ and $W(t_{(q,\bar{w},q')}, p_i) = \bar{w}[i]$ if $\bar{w}[i] \geq 0$. See Figure 3 for an example of the reduction. The initial marking corresponds to the initial weight vector in the natural way, and there is one extra token in the place p_{q_0} representing the current state of the automaton.

As before, it is easy to verify that the constructed Petri net has an infinite run if and only if the EL problem has a solution. The reduction clearly runs in polynomial time. □

Proof (of Theorem 2). Let $G_k = (Q_1, Q_2, q_0, \longrightarrow)$ be a k-weighted game and let $\bar{b}$ be a given upper bound vector for the GLU problem. We construct a $2k$-weighted game $G_{2k} = (Q_1 \uplus \{q_s\}, Q_2, q_s, \longrightarrow)$ where $q \xrightarrow{(\bar{w}[1],\bar{w}[2],\ldots,\bar{w}[k],-\bar{w}[1],-\bar{w}[2],\ldots,-\bar{w}[k])} q'$ in G_{2k} if and only if $q \xrightarrow{\bar{w}} q'$ in G_k. We moreover add the initial transition $q_s \xrightarrow{\bar{w}_0} q_0$ where $\bar{w}_0[i] = 0$ and $\bar{w}_0[k+i] = \bar{b}[i]$ for all i, $1 \leq i \leq k$. Figure 4 illustrates the construction on an example. Intuitively, every coordinate in the weight vector is duplicated and the duplicated coordinate gets initially the value from the vector $\bar{b}$, while the original coordinate is 0. It is now easy to verify that during any run in G_{2k} all its configurations $(q, \bar{v})$ satisfy the invariant $\bar{v}[i] + \bar{v}[k+i] = \bar{b}[i]$ for all i, $1 \leq i \leq k$.

The upper bound check is hence replaced with a lower bound on the duplicate coordinates and hence the GLU problem is reduced to GL and also to GLW (by using the weak upper bound vector $\bar{b}$), while the size of the weight vectors doubles. The reduction also clearly preserves the existential and universal variants of the problems. □

Proof (of Theorem 5). Let $G = (Q_1, Q_2, q_0, \longrightarrow)$ be a k-weighted game and let $\bar{b}$ be a given upper bound vector for the GLW problem. We will construct a corresponding k-weighted game $G' = (Q'_1, Q'_2, q_0, \longrightarrow)$ where $Q'_1 = Q_1 \cup \{q_1, q_2 \mid q \in Q_1\}$ and $Q'_2 = Q_2$ that simulates G.

As before we assume that the weak upper bound is $\bar{b} = (b, \ldots, b)$ and that b is represented using at most n bits, hence $0 \leq b < 2^n$. The new upper bound for G' is given as $\bar{b}' = (b', \ldots, b')$ where $b' = 2^n + b$ (in binary the most significant bit 1 is appended to the binary encoding of b).

Each transition $q \xrightarrow{\bar{w}} q'$ in G is simulated by a number of transitions in G' as seen in Figure 7. Moving from q to q_1 adds $\bar{w}$ to the accumulated weights of G' in exactly the same way as in G. In q_1 Player 1 has the opportunity to decrement independently all weight coordinates with an arbitrary value. The two last transitions from q_1 to q' make sure that in all coordinates all weights are no more than b, otherwise the upper bound $\bar{b}'$ is exceeded.

It is now clear that if Player 1 has a winning strategy in G, then it has a winning strategy also in G' by lowering all weights above b to exactly b in the state q_1. On the other hand, if Player 1 does not have a winning strategy in G, then it cannot win in G' either. This can be observed by the fact that Player 1 is forced to decrement all weights to at least b, and the player cannot benefit from decrementing them to any lower number as this makes the position of Player 1 in the weak upper bound game only worse.

Since the reduction is clearly in polynomial time and it adds only existential (Player 1) states, this concludes the proof. □

Proof (of Theorem 6). We shall first focus on the ELU problem. Notice that a parameterized ELU problem has an infinite run $(q_0, \bar{v}_0), (q_1, \bar{v}_1), \ldots$ where $\bar{0} \leq \bar{v}_i \leq \bar{b}$ for all i and some $\bar{b}$ if and only if there are two indices $j < k$ such that $(q_j, \bar{v}_j) = (q_k, \bar{v}_k)$. In other words, there is a cycle such that the accumulated weight on that cycle is exactly $\bar{0}$. A result in [13] shows that the existence of such zero-weight cycle is decidable in polynomial time.

Assume without loss of generality that the given weighted automaton contains only states reachable (while disregarding the weights) from the initial state q_0. It is now clear that if the weighted automaton contains a zero-weight cycle then the parameterized ELU problem has a solution by choosing an appropriate initial weight vector $\bar{v}_0$ and a sufficiently large upper bound $\bar{b}$ which enables us to execute the whole cycle plus reach the cycle from the initial pair $(q_0, \bar{v}_0)$. On the other hand, if there is no zero-weight cycle then the parameterized ELU does not have a solution, as for any choice of $\bar{v}_0$ and $\bar{b}$, every run will eventually violate either the lower bound or the upper bound.

By similar arguments, it is easy to see that a parameterized ELW problem has a solution if and only if the weighted automaton contains a nonnegative-weight cycle. To check for the existence of such a cycle in polynomial time we can use the trick described in [8]. We simply add to each state in the automaton a number of self-loops with weights $(-1, 0, \ldots, 0)$, $(0, -1, 0, \ldots, 0)$, $\ldots (0, \ldots, 0, -1)$ and then ask for the existence of a zero-weight cycle. □

Definition of Minsky two-counter machine. A *Minsky machine* with two nonnegative counters c_1 and c_2 is a sequence of labeled instructions $1 : \mathsf{inst}_1$; $2 : \mathsf{inst}_2$; $\ldots, n : \mathsf{inst}_n$ where $\mathsf{inst}_n = \mathsf{HALT}$ and each inst_i, $1 \leq i < n$, is of one of the following forms:

(Inc) i: c_j := c_j + 1; goto k

(Test-Dec) i: if c_j = 0 then goto k else (c_j := c_j - 1; goto ℓ)

for $j \in \{1, 2\}$ and $1 \leq k, \ell \leq n$. Instructions of type (Inc) are called *increment* instructions and of type (Test-Dec) are called *test and decrement* instructions.

A configuration is a triple (i, v_1, v_2) where i is the current instruction and v_1 and v_2 are the values of the counters c_1 and c_2 respectively. A computation step between configurations is defined in the natural way. If starting from the initial configuration $(1, 0, 0)$ the machine reaches the instruction HALT then we

say it *halts*, otherwise it *loops*. It is well known that the problem whether a given Minsky machine halts is undecidable [14].

Definition of k-weighted timed automaton. Let $\Phi(C)$ be the standard set of (diagonal-free) *clock constraints* over a finite set of *clocks* C given by conjunctions of constraints of the form $x \bowtie c$ with $x \in C$, $c \in \mathbb{Z}$, and $\bowtie$ any of the relations $\leq$, $<$, $=$, $>$, and $\geq$.

A k-weighted timed automaton is a tuple $T = (L, \ell_0, C, E, r, w)$, where L is a finite set of locations, $\ell_0 \in L$ is the initial location, C is a finite set of clocks, $E \subseteq L \times \Phi(C) \times 2^C \times L$ is a finite set of edges, and $r : L \to \mathbb{Z}^k$, $w : E \to \mathbb{Z}^k$ assign weight vectors to locations and edges.

Note that we allow weight updates on edges here; as shown in [3], this can have a significant influence on the complexity of the problems one wants to consider.

We also use the standard notation $v \models g$ for the fact that a *valuation* $v : C \to \mathbb{R}_{\geq 0}$ satisfies the clock constraint $g \in \Phi(C)$, $v + t$ for the valuation given by $(v + t)(x) = v(x) + t$, and $v[R]$ for the valuation with clocks in R reset to value 0.

The semantics of a k-weighted timed automaton is now given by a k-weighted automaton with states $Q = \{(\ell, v) \mid v \models I(\ell)\} \subseteq L \times \mathbb{R}^C_{\geq 0}$ and transitions

$(\ell, v) \xrightarrow{t \cdot r(\ell)} (\ell, v + t)$ for all $t' \in [0, t]$ (delay),

$(\ell, v) \xrightarrow{w(e)} (\ell', v')$ for all $e = (\ell, g, R, \ell') \in E$ s.t. $v \models g$ and $v' = v[R]$ (switch).

We recall the fact that weights on delay transitions may be non-integer real numbers; formally we have to change the definition of a k-weighted game to allow an infinite weighted transition relation $\longrightarrow \subseteq Q \times \mathbb{R}^k \times Q$. A *run* in a multiweighted timed automaton is a sequence of alternating switch and delay transitions in the corresponding multiweighted automaton.

Proof (of Theorem 8). We start by proving the case of ELU(2). The proof is by reduction from Minsky machines to multiweighted timed automata, based on the technique of the proof of Theorem 17 in [4]. We construct a one-clock 2-multiweighted timed automaton T that simulates a Minsky machine such that the Minsky machine loops if and only if T is a positive instance of the ELU(2) problem.

The values c_1, c_2 of the counters will be encoded by the accumulated weight vector $\bar{w} = (5 - 2^{-c_1}, 5 - 2^{-c_2})$ and T will start with an initial weight vector of $\bar{v}_0 = (4, 4)$, and the upper bound vector is $\bar{b} = (5, 5)$.

In order to simulate the instructions of the Minsky machine we now describe two different modules of T.

Increment and decrement: Figure 8 shows the general module used for incrementing and decrementing counter c_1; by interchanging the two weights one obtains the module for c_2. Note that the second component $\bar{w}[2]$ of the weight

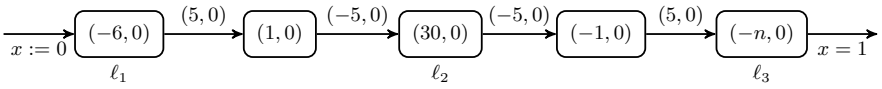


Fig. 8: The module for incrementing ($n = 3$) and decrementing ($n = 12$)

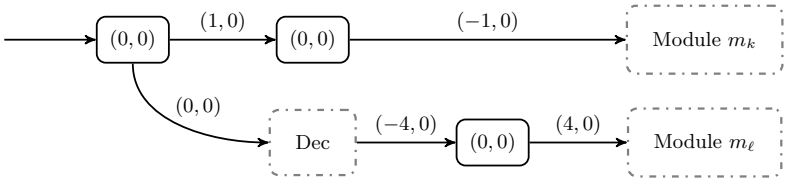


Fig. 9: The test-decrement module

vector is not changed in the module, and we assume that $\bar{w}[1] = 5 - e$ when entering the module and $0 \leq en \leq 30$. We now prove that when exiting the module, $\bar{w}[1] = 5 - \frac{en}{6}$.

Any legal run must decrease $\bar{w}[1]$ to value 0 while delaying in ℓ_1 (otherwise adding 5 to $\bar{w}[1]$ in the following transition exceeds the upper bound), hence the clock x has the value $\frac{5-e}{6}$ when leaving ℓ_1. We cannot delay in the next location, as this would exceed the upper bound, hence we arrive in ℓ_2 with $x = \frac{5-e}{6}$ and $\bar{w}[1] = 0$. We must delay in ℓ_2 until $\bar{w}[1]$ has the value 5, otherwise the following transition would exceed the lower bound, hence the delay in ℓ_2 is precisely $1/6$ time units. Location ℓ_3 is thus entered with $x = 1 - \frac{e}{6}$ and $\bar{w}[1] = 5$, and after delaying for $e/6$ time units, $\bar{w}[1] = 5 - \frac{en}{6}$.

Hence instantiating $n = 3$ converts an input of $\bar{w}[1] = 5 - e$ to $\bar{w}[1] = 5 - \frac{e}{2}$, thus incrementing counter c_1. Likewise, for $n = 12$ counter c_1 is decremented.

The test-decrement module: We have shown how to implement a module which increments a counter, so we miss to construct a module performing the instruction `if` $c_1 = 0$ `then goto` k `else (`c_1 `:= ` c_1 `- 1; goto` ℓ`)`. This module is displayed in Figure 9; the construction for the corresponding c_2 module is symmetric.

We now argue that the module acts as claimed. If $c_1 = 0$ when entering, i.e. $\bar{w}[1] = 4$, then the upper path can be taken, leading to Module m_k with counter value $c_1 = 0$ (and c_2 unchanged). On the other hand, attempting to take the lower path exits the Dec module with a value $\bar{w}[1] = 3$, hence the following transition leads to a violation of the lower bound.

If $c_1 \geq 1$, i.e. $\bar{w}[1] \geq 4.5$, when entering the module, then the $(1, 0)$ transition in the upper path will violate the upper bound. In the lower path, the Dec module is left with $\bar{w}[1] \geq 4$ and c_1 decreased by one, hence Module m_ℓ is entered with the correct c_1 value.

We have shown how to faithfully simulate a Minsky machine by a one-clock 2-multiweighted timed automaton such that the Minsky machine has an infinite computation if and only if the timed automaton has an infinite alternating run.

By undecidability of the halting problem for Minsky machines, this concludes the proof for the case of ELU(2).

For the case of EL(4) and ELW(4) we observe that the construction in the proof of Theorem 2 can be adapted also to multiweighted timed automata. Given a k-weighted timed automaton $T = (L, \ell_0, C, I, E, r, w)$ and an upper bound vector $\bar{b}$, we construct a $2k$-weighted timed automaton $T' = (L', \ell'_0, C, I', E', r', w')$ with $L' = L \uplus \{\ell'_0\}$, $I'(\ell) = I(\ell)$ for $\ell \in L$, $I'(\ell'_0) = (\bigwedge_{x \in C} x = 0)$, $E' = E \cup \{(\ell'_0, (\bigwedge_{x \in C} x = 0), \emptyset, \ell_0)\}$, $r'(\ell'_0) = \bar{0}$, and

$$r'(\ell) = (\bar{r}[1], \ldots, \bar{r}[k], -\bar{r}[1], \ldots, -\bar{r}[k]) \text{ for } \ell \in L \text{ and } \bar{r} = r(\ell),$$
$$w'(\ell, g, R, \ell') = (\bar{w}[1], \ldots, \bar{w}[k], -\bar{w}[1], \ldots, -\bar{w}[k])$$
$$\text{for } (\ell, g, R, \ell') \in E \text{ and } \bar{w} = w(\ell, g, R, \ell'),$$
$$w'(\ell'_0, g, R, \ell_0) = (0, \ldots, 0, \bar{b}[1], \ldots, \bar{b}[k]).$$

Then T is a positive instance of the ELU problem with an upper bound vector $\bar{b}$ if and only if T' is a positive instance of the EL or ELW (with weak upper bound vector $\bar{b}$) problems. The claim then follows from Theorem 8.

The results for the game versions of the problems follow from undecidability of GLU(1) [4] together with Theorem 2. □