Synthesis from LTL Specifications with Mean-Payoff Objectives

Aaron Bohy[1], Véronique Bruyère[1], Emmanuel Filiot[2], and Jean-François Raskin[3,*]

[1] Université de Mons
[2] Université Paris-Est Créteil
[3] Université Libre de Bruxelles

Abstract. The classical LTL synthesis problem is purely qualitative: the given LTL specification is realized or not by a reactive system. LTL is not expressive enough to formalize the correctness of reactive systems with respect to some quantitative aspects. This paper extends the *qualitative* LTL synthesis setting to a *quantitative* setting. The alphabet of actions is extended with a weight function ranging over the integer numbers. The value of an infinite word is the mean-payoff of the weights of its letters. The synthesis problem then amounts to automatically construct (if possible) a reactive system whose executions all satisfy a given LTL formula and have mean-payoff values greater than or equal to some given threshold. The latter problem is called LTL_{MP} synthesis and the LTL_{MP} realizability problem asks to check whether such a system exists. By reduction to two-player mean-payoff parity games, we first show that LTL_{MP} realizability is not more difficult than LTL realizability: it is 2ExpTime-Complete. While infinite memory strategies are required to realize LTL_{MP} specifications in general, we show that ϵ-optimality can be obtained with finite-memory strategies, for any $\epsilon > 0$. To obtain efficient algorithms in practice, we define a Safraless procedure to decide whether there exists a finite-memory strategy that realizes a given specification for some given threshold. This procedure is based on a reduction to two-player energy safety games which are in turn reduced to safety games. Finally, we show that those safety games can be solved efficiently by exploiting the structure of their state spaces and by using antichains as a symbolic data-structure. All our results extend to multi-dimensional weights. We have implemented an antichain-based procedure and we report on some promising experimental results.

1 Introduction

Formal specifications of reactive systems are usually expressed using formalisms like the linear temporal logic (LTL), the branching time temporal logic (CTL), or automata formalisms like Büchi automata. Those formalisms allow one to express *Boolean properties* in the sense that a reactive system either conforms to them, or violates them. Additionally to those qualitative formalisms, there is a clear need for another family of formalisms that are able to express *quantitative properties* of reactive systems. Abstractly, a quantitative property can be seen as a function that maps an execution of a reactive system to a numerical value. For example, in a client-server application, this

[*] Author supported by ERC Starting Grant (279499: inVEST).

numerical value could be the mean number of steps that separate the time at which a request has been emitted by a client and the time at which this request has been granted by the server along an execution. Quantitative properties are concerned with a large variety of aspects like quality of service, bandwidth, energy consumption,... But quantities are also useful to compare the merits of alternative solutions, e.g. we may prefer a solution in which the quality of service is high and the energy consumption is low. Currently, there is a large effort of the research community with the objective to lift the theory of formal verification and synthesis from the *qualitative world* to the richer *quantitative world* [15] (see related works for more details). In this paper, we consider mean-payoff and energy objectives. The alphabet of actions is extended with a weight function ranging over the integer numbers. A mean-payoff objective is a set of infinite words such that the mean value of the weights of their letters is greater than or equal to a given rational threshold [22], while an energy objective is parameterized by a non-negative initial energy level c_0 and contains all the words whose finite prefixes have a sum of weights greater than or equal to $-c_0$ [5].

In this paper, we participate to this research effort by providing theoretical complexity results, practical algorithmic solutions, and a tool for the automatic synthesis of reactive systems from *quantitative specifications* expressed in the linear time temporal logic LTL extended with (multi-dimensional) mean-payoff and (multi-dimensional) energy objectives. To illustrate our contributions, let us consider the following specification of a controller that should grant exclusive access to a resource to two clients.

Example 1. A client requests access to the resource by setting to true its request signal (r_1 for client 1 and r_2 for client 2), and the server grants those requests by setting to true the respective grant signal g_1 or g_2. We want to synthetize a server that eventually grants any client request, and that only grants one request at a time. This can be formalized in LTL as the conjunction of the three following formulas, where the signals in $I = \{r_1, r_2\}$ are controlled by the environment (the two clients), and the signals in $O = \{g_1, w_1, g_2, w_2\}$ are controlled by the server:

$$\phi_1 = \Box(r_1 \rightarrow \mathsf{X}(w_1 \mathsf{U} g_1)) \quad \phi_2 = \Box(r_2 \rightarrow \mathsf{X}(w_2 \mathsf{U} g_2)) \quad \phi_3 = \Box(\neg g_1 \vee \neg g_2)$$

Intuitively, ϕ_1 (resp. ϕ_2) specifies that any request of client 1 (resp. client 2) must be eventually granted, and in-between the waiting signal w_1 (resp. w_2) must be high. Formula ϕ_3 stands for mutual exclusion. Let $\phi = \phi_1 \wedge \phi_2 \wedge \phi_3$.

The formula ϕ is realizable. One possible strategy for the server is to alternatively assert w_2, g_1 and w_1, g_2, i.e. alternatively grant client 1 and client 2. While this strategy is formally correct, as it realizes the formula ϕ against all possible behaviors of the clients, it may not be the one that we expect. Indeed, we may prefer a solution that does not make unsollicited grants for example. Or, we may prefer a solution that gives, in case of request by both clients, some priority to client 2's request. In the later case, one elegant solution would be to associate a cost equal to 2 when w_2 is true and a cost equal to 1 when w_1 is true. This clearly will favor solutions that give priority to requests from client 2 over requests from client 1. We will develop other examples in the paper and describe the solutions that we obtain automatically with our algorithms.

Contributions – We now detail our contributions and give some hints about the proofs. In Section 2, we define the realizability problems for $\mathsf{LTL_{MP}}$ (LTL extended with mean-payoff objectives) and $\mathsf{LTL_{E}}$ (LTL extended with energy objectives). In Section 3, we show that, as for the LTL realizability problem, both the $\mathsf{LTL_{MP}}$ and $\mathsf{LTL_{E}}$ realizability problems are 2ExpTime-Complete. As the proof of those three results follow a similar structure, let us briefly recall how the 2ExpTime upper bound of the classical LTL realizability problem is established in [19]. The formula is turned into an equivalent nondeterministic Büchi automaton, which is then transformed into a deterministic parity automaton using Safra's construction. The latter automaton can be seen as a two-player parity game in which Player 1 wins if and only if the formula is realizable. For the $\mathsf{LTL_{MP}}$ (resp. $\mathsf{LTL_{E}}$) realizability problem, our construction follows the same structure, except that we go to a two-player parity game with an additional mean-payoff (resp. energy) objective. By a careful analysis of these two constructions, we build, on the basis of results in [8,11], solutions that provide the announced 2ExpTime upper bound.

Winning mean-payoff parity games may require infinite memory strategies, but there exist ϵ-optimal finite-memory strategies [11]. In contrast, for energy parity games, finite-memory optimal strategies always exist [8]. Those results transfer to $\mathsf{LTL_{MP}}$ (resp. $\mathsf{LTL_{E}}$) realizability problems thanks to their reduction to mean-payoff (resp. energy) parity games. Furthermore, we show that under finite-memory strategies, $\mathsf{LTL_{MP}}$ realizability is in fact equivalent to $\mathsf{LTL_{E}}$ realizability: a specification is MP-realizable under finite-memory strategies if and only if it is E-realizable, by simply shifting the weights of the signals by the threshold value. As finite-memory strategies are more interesting in practice, we thus concentrate on the $\mathsf{LTL_{E}}$ realizability problem in the rest of the paper.

Even if recent progresses have been made [21], Safra's construction is intricate and notoriously difficult to implement efficiently [1]. We develop in Section 4, following [17], a Safraless procedure for the $\mathsf{LTL_{E}}$ realizability problem, that is based on a reduction to a safety game, with the nice property to transform a quantitative objective into a simple qualitative objective. The main steps are as follows. *(1)* Instead of transforming an LTL formula into a deterministic parity automaton, we use a universal co-Büchi automaton as proposed in [17]. To deal with the energy objectives, we thus transform the formula into a universal co-Büchi energy automaton for some initial credit c_0, which requires that all runs on an input word w visit finitely many accepting states and the energy level of w is always positive starting from the credit c_0. *(2)* By strenghtening the co-Büchi condition into a K-co-Büchi condition as done in [20,14], where at most K accepting states can be visited by each run, we then go to an energy safety game. We show that for large enough value K and initial credit c_0, this reduction is complete. *(3)* Any energy safety game is equivalent to a safety game, as shown in [7].

In Section 5, our results are extended to the multi-dimensional case, i.e. tuples of weights. Finally, we discuss some implementation issues in Section 6. Our Safraless construction has two main advantages. *(1)* The search for winning strategies for $\mathsf{LTL_{E}}$ realizability can be incremental on K and c_0 (avoiding in practice the large theoretical bounds ensuring completeness). *(2)* The state space of the safety game can be partially ordered and solved by a backward fixpoint algorithm. Since the latter manipulates sets of states closed for this order, it can be made efficient and symbolic by working only on the antichain of their maximal elements. All the algorithms are implemented in our tool Acacia+ [3], and promising experimental results are reported in Section 6.

Due to lack of space, some proofs are omitted or just sketched. The full version is available at `http://arxiv.org/abs/1210.3539`.

Related Work – The LTL synthesis problem has been first solved in [19], Safraless approaches have been proposed in [16,17,20,14], and implemented in prototypes of tools [16,14,13,3]. All those works only treat plain qualitative LTL, and not the quantitative extensions considered in this article.

Mean-payoff games [22] and energy games [5,7], extensions with parity conditions [11,8,6], or multi-dimensions [10,12] have recently received a large attention from the research community. The use of such game formalisms has been advocated in [2] for specifying quantitative properties of reactive systems. Several among the motivations developed in [2] are similar that our motivations for considering quantitative extensions of LTL. All these related works make the assumption that the game graph is given explicitly, and not implicitly using an LTL formula, as in our case.

In [4], Boker et al. introduce extensions of linear and branching time temporal logics with operators to express constraints on values accumulated along the paths of a weighted Kripke structure. One of their extensions is similar to $\mathsf{LTL}_{\mathsf{MP}}$. However the authors of [4] only study the complexity of model-checking problems whereas we consider realizability and synthesis problems.

2 Problem Statement

Linear Temporal Logic – The formulas of linear temporal logic (LTL) are defined over a finite set P of atomic propositions. The syntax is given by the grammar:

$$\phi ::= p \mid \phi \vee \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi \qquad p \in P$$

LTL formulas ϕ are interpreted on infinite words $u \in (2^P)^\omega$ via a satisfaction relation $u \models \phi$ defined as usual [18]. Given ϕ, we let $[\![\phi]\!] = \{u \in (2^P)^\omega \mid u \models \phi\}$.

LTL *Realizability and Synthesis –* The realizability problem for LTL is best seen as a game between two players. Let ϕ be an LTL formula over the set $P = I \uplus O$ partitioned into I the set of *input signals* controlled by Player I (the environment), and O the set of *output signals* controlled by Player O (the controller). With this partition of P, we associate the three following alphabets: $\Sigma_P = 2^P$, $\Sigma_O = 2^O$, and $\Sigma_I = 2^I$. The realizability game is played in turns. Player O starts by giving $o_0 \in \Sigma_O$, Player I responds by giving $i_0 \in \Sigma_I$, then Player O gives $o_1 \in \Sigma_O$ and Player I responds by $i_1 \in \Sigma_I$, and so on. This game lasts forever and the outcome of the game is the infinite word $(o_0 \cup i_0)(o_1 \cup i_1)(o_2 \cup i_2) \cdots \in \Sigma_P^\omega$.

The players play according to *strategies*. A strategy for Player O is a mapping $\lambda_O : (\Sigma_O \Sigma_I)^* \to \Sigma_O$, while a strategy for Player I is a mapping $\lambda_I : (\Sigma_O \Sigma_I)^* \Sigma_O \to \Sigma_I$. The outcome of the strategies λ_O and λ_I is the word $\mathsf{Outcome}(\lambda_O, \lambda_I) = (o_0 \cup i_0)(o_1 \cup i_1) \ldots$ such that $o_0 = \lambda_O(\epsilon)$, $i_0 = \lambda_I(o_0)$ and for $k \geq 1$, $o_k = \lambda_O(o_0 i_0 \ldots o_{k-1} i_{k-1})$ and $i_k = \lambda_I(o_0 i_0 \ldots o_{k-1} i_{k-1} o_k)$. We denote by $\mathsf{Outcome}(\lambda_O)$ the set of all outcomes

$\mathsf{Outcome}(\lambda_O, \lambda_I)$ with λ_I any strategy of Player I. We let Π_O (resp. Π_I) be the set of strategies for Player O (resp. Player I).

Given an LTL formula ϕ (the specification), the LTL *realizability problem* is to decide whether there exists $\lambda_O \in \Pi_O$ such that for all $\lambda_I \in \Pi_I$, $\mathsf{Outcome}(\lambda_O, \lambda_I) \models \phi$. If such a *winning* strategy exists, we say that the specification ϕ is *realizable*. The LTL *synthesis problem* asks to produce a strategy λ_O that realizes ϕ, when it is realizable.

Moore Machines – It is known that LTL realizability is 2ExpTime-Complete and that finite-memory strategies suffice to witness realizability [19]. A strategy $\lambda_O \in \Pi_O$ is *finite-memory* if there exists a right-congruence $\sim$ on $(\Sigma_O \Sigma_I)^*$ of finite index such that $\lambda_O(u) {=} \lambda_O(u')$ for all $u \sim u'$. It is equivalent to say that it can be described by a *Moore machine* $\mathcal{M}$, i.e. a finite deterministic state machine with output [19]. If the machine $\mathcal{M}$ describes λ_O, then $\mathsf{Outcome}(\lambda_O)$ is called the *language of* $\mathcal{M}$, denoted by $L(\mathcal{M})$. The memory size of the strategy is the index of $\sim$.

Theorem 1 ([19]). *The* LTL *realizability problem is 2ExpTime-Complete and any realizable formula is realizable by a finite-memory strategy with memory size* $2^{2^{O(|\phi| \log(|\phi|))}}$.

$\mathsf{LTL_{MP}}$ *Realizability and Synthesis –* Consider a finite set P partitioned as $I \uplus O$. Let $\mathsf{Lit}(P)$ be the set $\{p \mid p \in P\} \cup \{\neg p \mid p \in P\}$ of literals over P, and let $w : \mathsf{Lit}(P) \to \mathbb{Z}$ be a *weight function* where positive numbers represent rewards[1]. For all $S \in \{I, O\}$, this function is extended to Σ_S by: $w(\sigma) = \Sigma_{p \in \sigma} w(p) + \Sigma_{p \in S \setminus \{\sigma\}} w(\neg p)$ for $\sigma \in \Sigma_S$. It can also be extended to Σ_P as $w(o \cup i) = w(o) + w(i)$ for all $o \in \Sigma_O$ and $i \in \Sigma_I$. In the sequel, we denote by $\langle P, w \rangle$ the pair given by the finite set P and the weight function w over $\mathsf{Lit}(P)$; we also use the weighted alphabet $\langle \Sigma_P, w \rangle$.

Consider an LTL formula ϕ over $\langle P, w \rangle$ and an outcome $u = (o_0 \cup i_0)(o_1 \cup i_1) \dots \in \Sigma_P^\omega$ produced by Players I and O. We associate a *value* $\mathsf{Val}(u)$ with u that captures the two objectives of Player O of both satisfying ϕ and achieving a mean-payoff objective. For each $n \geq 0$, let $u(n)$ be the prefix of u of length n. We define the *energy level* of $u(n)$ as $\mathsf{EL}(u(n)) = \sum_{k=0}^{n-1} w(o_k) + w(i_k)$. We then assign to u a *mean-payoff value* equal to $\mathsf{MP}(u) = \liminf_{n \to \infty} \frac{1}{n} \mathsf{EL}(u(n))$. Finally we define the value of u as $\mathsf{Val}(u) = \mathsf{MP}(u)$ if $u \models \phi$, and $\mathsf{Val}(u) = -\infty$ otherwise.

Given an LTL formula ϕ over $\langle P, w \rangle$ and a threshold $\nu {\in} \mathbb{Q}$, the $\mathsf{LTL_{MP}}$ *realizability problem (resp.* $\mathsf{LTL_{MP}}$ *realizability problem under finite memory)* asks to decide whether there exists a strategy (resp. finite-memory strategy) λ_O of Player O such that for all strategies $\lambda_I \in \Pi_I$, $\mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)) \geq \nu$, in which case we say that ϕ is MP-*realizable (resp.* MP*-realizable under finite memory)*. The $\mathsf{LTL_{MP}}$ *synthesis problem* is to produce such a winning strategy λ_O. So the aim is to achieve two objectives: *(i)* realizing ϕ, *(ii)* having a long-run average reward greater than the given threshold.

Optimality – Given ϕ an LTL formula over $\langle P, w \rangle$, the *optimal value* (for Player O) is defined as $\nu_\phi = \sup_{\lambda_O \in \Pi_O} \inf_{\lambda_I \in \Pi_I} \mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I))$. For a real-valued $\epsilon \geq 0$, a strategy λ_O of Player O is ϵ*-optimal* if $\mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)) \geq \nu_\phi - \epsilon$ against all

[1] We use weights at several places of this paper. In some statements and proofs, we take the freedom to use *rational* weights as it is equivalent up to rescaling. However we always assume that weights are integers encoded in binary for complexity results.

strategies λ_I of Player I. It is *optimal* if it is ϵ-optimal with $\epsilon = 0$. Notice that ν_ϕ is equal to $-\infty$ if Player O cannot realize ϕ.

Example 2. Let us come back to Example 1 of a client-server system with two clients sharing a resource. The specification have been formalized by an LTL formula ϕ over the alphabet $P{=}I \uplus O$, with $I{=}\{r_1, r_2\}$, $O{=}\{g_1, w_1, g_2, w_2\}$. Suppose that we want to add the following constraints: client 2's requests take the priority over client 1's requests, but client 1's should still be eventually granted. Moreover, we would like to keep minimal the delay between requests and grants. This latter requirement has more the flavor of an optimality criterion and is best modeled using a weight function and a mean-payoff objective. To this end, we impose penalties to the waiting signals w_1, w_2, with a larger penalty to w_2 than to w_1. We thus use the following weight function $w : \mathsf{Lit}(P) \to \mathbb{Z}$: $w(w_1) = -1, w(w_2) = -2$ and $w(l) = 0, \forall l \notin \{w_1, w_2\}$.

One optimal strategy for the server is as follows: it almost always grants the resource to client 2 immediately after r_2 is set to true by client 2, and with a decreasing frequency grants request r_1 emitted by client 1. Such a server ensures a mean-payoff value equal to -1 against the most demanding behavior of the clients (where they are constantly requesting the shared resource). Such a strategy requires the server to use an infinite memory as it has to grant client 1 with an infinitely decreasing frequency. Note that a server that would grant client 1 in such a way without the presence of requests by client 1 would still be optimal. No finite memory server can be optimal. Indeed, if the server is allowed to count only up to a fixed positive integer $k \in \mathbb{N}$, then the best that it can do is : grant immediatly any request by client 2 if the last ungranted request of client 1 has been emitted less than k steps in the past, otherwise grant the request of client 1. The mean-payoff value of this solution, in the worst-case (when the two clients always emit their respective request) is equal to $-(1 + \frac{1}{k})$. So, even if finite memory cannot be optimal, we can devise a finite-memory strategy that is ϵ-optimal for any $\epsilon > 0$.

$\mathsf{LTL_E}$ *Realizability and Synthesis –* For the proofs of this paper, we need to consider realizability and synthesis with *energy* (instead of mean-payoff) objectives. With the same notations as before, the $\mathsf{LTL_E}$ *realizability problem* is to decide whether ϕ is E-*realizable*, that is, whether there exists $\lambda_O \in \Pi_O$ and $c_0 \in \mathbb{N}$ such that for all $\lambda_I \in \Pi_I$, *(i)* $u = \mathsf{Outcome}(\lambda_O, \lambda_I) \models \phi$, *(ii)* $\forall n \geq 0, c_0 + \mathsf{EL}(u(n)) \geq 0$. We thus ask if there exists an *initial credit* c_0 such that the energy level of each prefix $u(n)$ remains positive. When ϕ is E-realizable, the $\mathsf{LTL_E}$ *synthesis problem* is to produce such a winning strategy λ_O. Finally, we define the *minimum initial credit* as the least value of c_0 for which ϕ is E-realizable. A strategy λ_O is *optimal* if it is winning for the minimum initial credit.

3 Computational Complexity of the $\mathsf{LTL_{MP}}$ Realizability Problem

In this section, we solve the $\mathsf{LTL_{MP}}$ realizability problem, and we establish its complexity. Our solution relies on a reduction to a mean-payoff parity game. The same complexity result holds for the $\mathsf{LTL_E}$ realizability problem.

Theorem 2. *The* $\mathsf{LTL_{MP}}$ *realizability problem is 2ExpTime-Complete.*

Before proving this result, we recall useful notions on game graphs.

Game Graphs – A *game graph* $G = (S, s_0, E)$ consists of a finite set $S = S_1 \uplus S_2$ partitioned into S_1 the states of Player 1, and S_2 the states of Player 2, an initial state s_0, and a set $E \subseteq S \times S$ of edges such that for all $s \in S$, there exists a state $s' \in S$ such that $(s, s') \in E$. A game on G starts from s_0 and is played in rounds as follows. If the game is in a state of S_1, then Player 1 chooses the successor state among the set of outgoing edges; otherwise Player 2 chooses the successor state. Such a game results in an infinite path $\rho = s_0 s_1 \dots s_n \dots$ (called a *play*), whose prefix $s_0 s_1 \dots s_n$ is denoted by $\rho(n)$. We denote by $\mathsf{Plays}(G)$ the set of all plays in G and by $\mathsf{Pref}(G)$ the set of all prefixes of plays in G. A *turn-based* game is a game graph G such that $E \subseteq (S_1 \times S_2) \cup (S_2 \times S_1)$, meaning that each game is played in rounds alternatively by Player 1 and Player 2.

Objectives – An *objective* for G is a set $\Omega \subseteq S^\omega$. Let $p : S \rightarrow \mathbb{N}$ be a *priority function* and $w : E \rightarrow \mathbb{Z}$ be a *weight function* where positive weights represent rewards. The *energy level* of a prefix $\gamma = s_0 s_1 \dots s_n$ of a play is $\mathsf{EL}_G(\gamma) = \sum_{i=0}^{n-1} w(s_i, s_{i+1})$, and the *mean-payoff value* of a play $\rho = s_0 s_1 \dots s_n \dots$ is $\mathsf{MP}_G(\rho) = \liminf_{n\to\infty} \frac{1}{n} \cdot \mathsf{EL}_G(\rho(n))$.[2] Given a play ρ, we denote by $\mathsf{Inf}(\rho)$ the set of states $s \in S$ that appear infinitely often in ρ. The following objectives Ω are considered in the sequel:

- *Safety objective.* Given a set $\alpha \subseteq S$, the safety objective is defined as $\mathsf{Safety}_G(\alpha) = \mathsf{Plays}(G) \cap \alpha^\omega$.
- *Parity objective.* The parity objective is defined as $\mathsf{Parity}_G(p) = \{\rho \in \mathsf{Plays}(G) \mid \min\{p(s) \mid s \in \mathsf{Inf}(\rho)\}$ is even$\}$.
- *Energy objective.* Given an initial credit $c_0 \in \mathbb{N}$, the energy objective is defined as $\mathsf{PosEnergy}_G(c_0) = \{\rho \in \mathsf{Plays}(G) \mid \forall n \geq 0 : c_0 + \mathsf{EL}_G(\rho(n)) \geq 0\}$.
- *Mean-payoff objective.* Given a threshold $\nu \in \mathbb{Q}$, the mean-payoff objective is defined as $\mathsf{MeanPayoff}_G(\nu) = \{\rho \in \mathsf{Plays}(G) \mid \mathsf{MP}_G(\rho) \geq \nu\}$.
- *Combined objective.* The *energy safety* objective $\mathsf{PosEnergy}_G(c_0) \cap \mathsf{Safety}_G(\alpha)$ (resp. *energy parity* objective $\mathsf{PosEnergy}_G(c_0) \cap \mathsf{Parity}_G(p)$, *mean-payoff parity* objective $\mathsf{MeanPayoff}_G(\nu) \cap \mathsf{Parity}_G(p)$) combines the requirements of energy and safety (resp. energy and parity, energy and mean-payoff) objectives.

When an objective Ω is imposed on G, we say that G is an Ω *game*. For instance, if Ω is an energy parity objective, we say that $\langle G, w, p\rangle$ is an *energy parity game*, aso.

Strategies – Given a game graph G, a *strategy* for Player 1 is a function $\lambda_1 : S^* S_1 \rightarrow S$ such that $(s, \lambda_1(\gamma \cdot s)) \in E$ for all $\gamma \in S^*$ and $s \in S_1$. A play $\rho = s_0 s_1 \dots s_n \dots$ starting from the initial state s_0 is compatible with λ_1 if for all $k \geq 0$ such that $s_k \in S_1$ we have $s_{k+1} = \lambda_1(\rho(k))$. Strategies and play compatibility are defined symmetrically for Player 2. The set of strategies of Player i is denoted by Π_i, i=$1, 2$. We denote by $\mathsf{Outcome}_G(\lambda_1, \lambda_2)$ the play from s_0, compatible with λ_1 and λ_2. We let $\mathsf{Outcome}_G(\lambda_1) = \{\mathsf{Outcome}_G(\lambda_1, \lambda_2) \mid \lambda_2 {\in} \Pi_2\}$. A strategy $\lambda_1 {\in} \Pi_1$ is *winning* for an objective Ω if $\mathsf{Outcome}_G(\lambda_1) \subseteq \Omega$. We also say that λ_1 is winning in the Ω game G.

A strategy λ_1 of Player 1 is *finite-memory* if there exists a right-congruence $\sim$ on $\mathsf{Pref}(G)$ with finite index such that $\lambda_1(\gamma \cdot s_1) = \lambda_1(\gamma' \cdot s_1)$ for all $\gamma \sim \gamma'$ and $s_1 \in S_1$. The *size* of the memory is the index of $\sim$.

[2] Notation EL, MP and $\mathsf{Outcome}$ is here used with the index G to avoid any confusion with the same notation introduced in the previous section.

Optimal Value and ϵ-Optimality – Let us turn to mean-payoff parity games $\langle G, w, p\rangle$. With each play $\rho \in \mathsf{Plays}(G)$, we associate a *value* $\mathsf{Val}_G(\rho)$ defined as follows:

$$\mathsf{Val}_G(\rho) = \begin{cases} \mathsf{MP}_G(\rho) & \text{if } \rho \in \mathsf{Parity}_G(p) \\ -\infty & \text{otherwise.} \end{cases}$$

We define $\nu_G = \sup_{\lambda_1 \in \Pi_1} \inf_{\lambda_2 \in \Pi_2} \mathsf{Val}_G(\mathsf{Outcome}_G(\lambda_1, \lambda_2))$ as the *optimal value* for Player 1. For a real-valued $\epsilon \geq 0$, a strategy $\lambda_1 \in \Pi_1$ is *ϵ-optimal* if we have $\mathsf{Val}_G(\mathsf{Outcome}_G(\lambda_1, \lambda_2)) \geq \nu_G - \epsilon$ for all strategies $\lambda_2 \in \Pi_2$. It is *optimal* if it is ϵ-optimal with $\epsilon = 0$. If Player 1 cannot achieve the parity objective, then $\nu_G = -\infty$, otherwise optimal strategies exist [11] and ν_G is the largest threshold ν for which Player 1 can hope to achieve $\mathsf{MeanPayoff}_G(\nu)$.

Theorem 3 ([11,6,12]). *The optimal value of a mean-payoff parity game $\langle G, w, p\rangle$ can be computed in time $O(|E| \cdot |S|^{d+2} \cdot W)$, where $|E|$ (resp. $|S|$) is the number of edges (resp. states) of G, d is the number of priorities of p, and W is the largest absolute weight used by w. When $\nu_G \neq -\infty$, optimal strategies for Player 1 may require infinite memory; however for all $\epsilon > 0$, Player 1 has a finite-memory ϵ-optimal strategy.*

Proof (of Theorem 2). The classical realizability procedure for plain LTL first transforms the LTL formula into a non-deterministic Büchi automaton and then into a deterministic parity automaton. This deterministic automaton directly defines a parity game in which Player 1 has a strategy iff Player O has a strategy to realize the LTL specification. For a $\mathsf{LTL}_{\mathsf{MP}}$ specification ϕ, we follow the same path but extend the parity game into a mean-payoff parity game using the weight function w. It can be shown that the game we obtain has the following size: $2^{2^{O(|\phi| \log |\phi|)}}$ states and $2^{O(|\phi|)}$ priorities. By Theorem 3, we get the 2ExpTime upper bound for $\mathsf{LTL}_{\mathsf{MP}}$ realizability. The lower bound is a direct consequence of 2ExpTime-hardness of (qualitative) LTL realizability. $\square$

Based on Theorem 3 and the proof of Theorem 2 we get the following results on ϵ-optimality and finite-memory strategies:

Corollary 4. *Let ϕ be an LTL formula. If ϕ is MP-realizable, then for all $\epsilon > 0$, Player O has an ϵ-optimal winning strategy that is finite-memory, that is*

$$\nu_\phi = \sup_{\substack{\lambda_O \in \Pi_O \\ \lambda_O \text{ finite-memory}}} \inf_{\lambda_I \in \Pi_I} \mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)).$$

Motivated by this result, we focus on finite-memory strategies in the sequel.

Solution to the LTL_E Realizability Problem – We solve the LTL_E realizability problem with a reduction to energy parity games for which the following theorem holds:

Theorem 5 ([8]). *Whether there exist an initial credit c_0 and a winning strategy for Player 1 in a given energy parity game $\langle G, w, p\rangle$ for c_0 can be decided in time $O(|E| \cdot d \cdot |S|^{d+3} \cdot W)$. Moreover if Player 1 wins, then he has a finite-memory winning strategy with a memory size bounded by $4 \cdot |S| \cdot d \cdot W$.*

As for $\mathsf{LTL_{MP}}$, one can reduce $\mathsf{LTL_E}$ realizability to energy parity games and show that $\mathsf{LTL_E}$ realizability is 2ExpTime-Complete based on Theorem 5.

Theorem 6. *The* $\mathsf{LTL_E}$ *realizability problem is 2ExpTime-Complete. Moreover, if a formula ϕ over $\langle P, w\rangle$ is* E*-realizable, then it is* E*-realizable by a finite-memory strategy with a memory size at most doubly-exponential in the size of the input, i.e. the* LTL *formula and the function w (with weights encoded in binary).*

The constructions proposed in Theorems 2 and 6 can be easily extended to the more general case where the weights assigned to executions are given by a deterministic weighted automaton, as proposed in [9], instead of a weight function w over $\mathsf{Lit}(P)$ as done here. Indeed, given an LTL formula ϕ and a deterministic weighted automaton $\mathcal{A}$, we first construct from ϕ a deterministic parity automaton and then take the synchronized product with $\mathcal{A}$. Finally this product can be turned into a mean-payoff (resp. energy) parity game.

4 Safraless Algorithm

In the previous section, we have proposed an algorithm for solving the $\mathsf{LTL_{MP}}$ realizability of a given LTL formula ϕ, which is based on a reduction to a mean-payoff parity game denoted by G_ϕ. This algorithm has two main drawbacks. First, it requires the use of Safra's construction to get a deterministic parity automaton $\mathcal{A}_\phi$ such that $L(\mathcal{A}_\phi) = [\![\phi]\!]$, a construction which is resistant to efficient implementations [1]. Second, strategies for the game G_ϕ may require infinite memory (for the threshold ν_{G_ϕ}, see Theorem 3). This is also the case for the $\mathsf{LTL_{MP}}$ realizability problem, as illustrated by Example 2. In this section, we show how to circumvent these two drawbacks.

The second drawback has been already partially solved by Corollary 4, when the threshold given for the $\mathsf{LTL_{MP}}$-realizability is the optimal value ν_ϕ. Indeed it states the existence of finite-memory winning strategies for the thresholds $\nu_\phi - \epsilon$, for all $\epsilon > 0$. We here show that we can go further by translating the $\mathsf{LTL_{MP}}$ realizability problem under finite memory into an $\mathsf{LTL_E}$ realizability problem, and conversely, by shifting the weights by the threshold value [8]:

Theorem 7. *An* LTL *formula ϕ over a weighted alphabet $\langle P, w\rangle$ is* MP*-realizable under finite memory for a threshold $\nu \in \mathbb{Q}$ iff ϕ over the weighted alphabet $\langle P, w - \nu\rangle$ is* E*-realizable.*

It is important to notice that when we want to synthesize ϵ-optimal strategies for $\mathsf{LTL_{MP}}$ by reduction to $\mathsf{LTL_E}$, the memory size of the strategy increases as ϵ decreases. Indeed, if $\epsilon = \frac{a}{b}$, then the weight function (for $\mathsf{LTL_E}$ realizability) must be multiplied by b in a way to have integer weights (see footnote 1). The largest absolute weight W is thus also multiplied by b.

To avoid the Safra's construction needed to obtain a deterministic parity automaton for the underlying LTL formula, we adapt a Safraless construction proposed in [20,14] for the LTL synthesis problem, in a way to deal with weights and efficiently solve the $\mathsf{LTL_E}$ synthesis problem. Instead of constructing a mean-payoff parity game from a deterministic parity automaton, we propose a reduction to a safety game. In this aim, we need to define the notion of energy automaton.

Energy Automata – Let $\langle P, w\rangle$ with P a finite set of signals and w a weight function over $\mathsf{Lit}(P)$. We are going to recall several notions of automata on infinite words over Σ_P and introduce the related notion of energy automata over the weighted alphabet $\langle \Sigma_P, w\rangle$. An *automaton* $\mathcal{A}$ over the alphabet Σ_P is a tuple $(\Sigma_P, Q, q_0, \alpha, \delta)$ such that Q is a finite set of states, $q_0 \in Q$ is the initial state, $\alpha \subseteq Q$ is a set of final states and $\delta : Q \times \Sigma_P \to 2^Q$ is a transition function. We say that $\mathcal{A}$ is *deterministic* if $\forall q \in Q, \forall \sigma \in \Sigma_P, |\delta(q,\sigma)| \leq 1$. It is *complete* if $\forall q \in Q, \forall \sigma \in \Sigma_P, \delta(q,\sigma) \neq \varnothing$.

A *run* of $\mathcal{A}$ on a word $u = \sigma_0\sigma_1 \cdots \in \Sigma_P^\omega$ is an infinite sequence of states $\rho = \rho_0\rho_1 \cdots \in Q^\omega$ such that $\rho_0 = q_0$ and $\forall k \geq 0, \rho_{k+1} \in \delta(\rho_k, \sigma_k)$. We denote by $\mathsf{Runs}_{\mathcal{A}}(u)$ the set of runs of $\mathcal{A}$ on u, and by $\mathsf{Visit}(\rho, q)$ the number of times the state q occurs along the run ρ. We consider the following acceptance conditions:

Non-deterministic Büchi:	$\exists \rho \in \mathsf{Runs}_{\mathcal{A}}(u), \exists q \in \alpha, \mathsf{Visit}(\rho, q) = \infty$
Universal co-Büchi:	$\forall \rho \in \mathsf{Runs}_{\mathcal{A}}(u), \forall q \in \alpha, \mathsf{Visit}(\rho, q) < \infty$
Universal K-co-Büchi:	$\forall \rho \in \mathsf{Runs}_{\mathcal{A}}(u), \sum_{q\in\alpha} \mathsf{Visit}(\rho, q) \leq K.$

A word $u \in \Sigma_P^\omega$ is *accepted* by a *non-deterministic Büchi automaton* (NB) $\mathcal{A}$ if u satisfies the non-deterministic Büchi acceptance condition. We denote by $L_{\text{nb}}(\mathcal{A})$ the set of words accepted by $\mathcal{A}$. Similarly we have the notion of *universal co-Büchi automaton* (UCB) $\mathcal{A}$ (resp. *universal K-co-Büchi automaton* ($\mathsf{U}K\mathsf{CB}$) $\langle \mathcal{A}, K\rangle$) and the set $L_{\text{ucb}}(\mathcal{A})$ (resp. $L_{\text{ucb},K}(\mathcal{A})$) of accepted words.

We now introduce energy automata. Let $\mathcal{A}$ be a NB over the alphabet Σ_P. The related *energy non-deterministic Büchi automaton* (eNB) $\mathcal{A}^w$ is over the weighted alphabet $\langle \Sigma_P, w\rangle$ and has the same structure as $\mathcal{A}$. Given an initial credit $c_0 \in \mathbb{N}$, a word u is *accepted* by $\mathcal{A}^w$ if *(i)* u satisfies the non-deterministic Büchi acceptance condition and *(ii)* $\forall n \geq 0, c_0 + \mathsf{EL}(u(n)) \geq 0$. We denote by $L_{\text{nb}}(\mathcal{A}^w, c_0)$ the set of words accepted by $\mathcal{A}^w$ with the given initial credit c_0. We also have the notions of *energy universal co-Büchi automaton* (eUCB) $\mathcal{A}^w$ and *energy universal K-co-Büchi automaton* ($\mathsf{eU}K\mathsf{CB}$) $\langle \mathcal{A}^w, K\rangle$, and the related sets $L_{\text{ucb}}(\mathcal{A}^w, c_0)$ and $L_{\text{ucb},K}(\mathcal{A}^w, c_0)$. Notice that if $K \leq K'$ and $c_0 \leq c_0'$, then $L_{\text{ucb},K}(\mathcal{A}^w, c_0) \subseteq L_{\text{ucb},K'}(\mathcal{A}^w, c_0')$.

The interest of $\mathsf{U}K\mathsf{CB}$ is that they can be determinized with the subset construction extended with counters [14,20]. This construction also holds for $\mathsf{eU}K\mathsf{CB}$ by using counting functions F. Intuitively, for all states q of $\mathcal{A}^w$, with $F(q)$ we count (up to $K+1$) the maximal number of accepting states which have been visited by runs ending in q. The counter $F(q)$ is equal to -1 when no run ends in q. The final states are counting functions F such that $F(q)>K$ for some state q (accepted runs avoid such F).

It results in a deterministic automaton that we denote $\det(\mathcal{A}^w, K)$ and which has the following properties:

Proposition 8. *Let $K\in\mathbb{N}$ and $\langle \mathcal{A}^w, K\rangle$ be an* $\mathsf{eU}K\mathsf{CB}$. *Then $\det(\mathcal{A}^w, K)$ is a deterministic and complete energy automaton such that $L_{ucb,0}(\det(\mathcal{A}^w, K), c_0) = L_{ucb,K}(\mathcal{A}^w, c_0)$ for all $c_0 \in \mathbb{N}$.*

Our Safraless solution relies on the following theorem:

Theorem 9. *Let ϕ be an* LTL *formula over $\langle P, w_P\rangle$. Let $\langle G_\phi, w, p\rangle$ be the associated energy parity game with $|S|$ being its the number of states, d its number of priorities and W its largest absolute weight. Let $\mathcal{A}$ be a* UCB *with n states such that $L_{ucb}(\mathcal{A}) = [\![\phi]\!]$.*

Let $K = 4 \cdot n \cdot |S|^2 \cdot d \cdot W$ and $C = K \cdot W$. Then ϕ is E*-realizable iff there exists a Moore machine $\mathcal{M}$ such that $L(\mathcal{M}) \subseteq L_{ucb,K}(\mathcal{A}^w, C)$.*

Proof. Theorem 6 tells us that ϕ is E-realizable iff there exists a Moore machine $\mathcal{M}$ such that $L(\mathcal{M}) \subseteq L_{\text{ucb}}(\mathcal{A}^w, c_0)$ for some $c_0 \geq 0$ and $|\mathcal{M}| = 4 \cdot |S|^2 \cdot d \cdot W$. Consider now the product of $\mathcal{M}$ and $\mathcal{A}^w$: in any accessible cycle of this product, there is no accepting state of $\mathcal{A}^w$ (as shown similarly for the qualitative case [14]) and the sum of the weights must be positive. The length of a path reaching such a cycle is at most $n \cdot |\mathcal{M}|$, therefore one gets $L(\mathcal{M}) \subseteq L_{\text{ucb},n\cdot|\mathcal{M}|}(\mathcal{A}^w, n \cdot |\mathcal{M}| \cdot W)$. □

As we have seen before, the eU*K*CB $\mathcal{A}^w$ can be easily determinized and thus converted into an energy safety objective. By memorizing the energy levels up to C [7], this energy safety objective can be converted into a safety objective, and so we get:

Theorem 10. *Let ϕ be an* LTL *formula. Then one can construct a safety game in which Player 1 has a winning strategy iff ϕ is* E*-realizable.*

5 Extension to Multi-dimensional Weights

Multi-Dimensional $\mathsf{LTL_{MP}}$ *and* $\mathsf{LTL_E}$ *Realizability Problems –* The $\mathsf{LTL_{MP}}$ and $\mathsf{LTL_E}$ realizability problems can be naturally extended to multi-dimensional weights. Given P, we define a weight function $w : \mathsf{Lit}(P) \to \mathbb{Z}^m$, for some dimension $m \geq 1$. The concepts of energy level EL, mean-payoff value MP, and value Val are defined similarly. Given an LTL formula ϕ over $\langle P, w\rangle$ and a threshold $\nu \in \mathbb{Q}^m$, the *multi-dimensional* $\mathsf{LTL_{MP}}$ *realizability problem under finite memory* asks to decide whether there exists a Player O's finite-memory strategy λ_O such that $\mathsf{Val}(\mathsf{Outcome}(\lambda_O, \lambda_I)) \geq^3 \nu$ against all strategies $\lambda_I \in \Pi_I$. The *multi-dimensional* $\mathsf{LTL_E}$ *realizability problem* asks to decide whether there exists $\lambda_O \in \Pi_O$ and an initial credit $c_0 \in \mathbb{N}^m$ such that for all $\lambda_I \in \Pi_I$, *(i)* $u = \mathsf{Outcome}(\lambda_O, \lambda_I) \models \phi$, *(ii)* $\forall n \geq 0,\ c_0 + \mathsf{EL}(u(n)) \geq (0, \ldots, 0)$.

Computational Complexity – The 2ExpTime-completeness of the $\mathsf{LTL_{MP}}$ and $\mathsf{LTL_E}$ realizability problems have been stated in Theorem 2 and 6 in one dimension. In the multi-dimensional case, we have the next result.

Theorem 11. *The multi-dimensional* $\mathsf{LTL_{MP}}$ *realizability problem under finite memory and the multi-dimensional* $\mathsf{LTL_E}$ *realizability problem are in co-N2ExpTime.*

Proof. As for the one-dimensional case, $\mathsf{LTL_{MP}}$ realizability problem under finite memory and the multi-dimensional $\mathsf{LTL_E}$ realizability problem are inter-reducible by substracting the threshold to the weight values. So let us focus on $\mathsf{LTL_E}$ realizability. From the LTL formula we follow the same path as the one-dimensional case by constructing an equivalent deterministic parity automaton, that can be seen as a parity game. We add multi-weights to this game and so the $\mathsf{LTL_E}$ realizability problem amounts to solve a multi-energy parity game. Such games have been studied in [12], where it is shown how to remove the parity condition by adding extra dimensions in the game. This leads

[3] With $a \geq b$, we mean $a_i \geq b_i$ for all i, $1 \leq i \leq m$.

to resolving a multi-energy game, which can be done with a co-NPTime procedure, as shown in [10]. As this procedure executes on a doubly exponential game, we get the co-N2ExpTime upper bound. □

The Safraless procedure that we propose in one dimension can be extended to this multi-dimensional setting using recent results obtained in [12].

6 Implementation and Experiments

In the previous sections, in one or several dimensions, we have shown how to reduce the $\mathsf{LTL}_{\mathsf{MP}}$ under finite memory and $\mathsf{LTL}_{\mathsf{E}}$ realizability problems to safety games. We first discuss how antichain techniques can be used to symbolically solve those safety games. This approach has been implemented in our tool Acacia+. We then briefly present this tool and give some experimental results.

Antichain-Based Algorithm – Safety games with the objective $\mathsf{Safety}_G(\alpha)$ can be solved backwardly by computing the fixpoint of the following sequence: $W_0{=}\alpha$ and for all $k \geq 0$, $W_{k+1}{=}W_k \cap \{\{s{\in}S_1 \mid \exists(s,s'){\in}E, s' \in W_k\} \cup \{s{\in}S_2 \mid \forall(s,s'){\in}E, s' \in W_k\}\}$.

Therefore one needs to manipulate sets of states. The states of the safety game in our Safraless procedure are tuples (F, c) where F is a counting function as described before Proposition 8 and c is an energy level. They can be ordered as follows: $(F_1, c_1) \preceq (F_2, c_2)$ iff $F_1(q) \leq F_2(q)$ for all automata state q and $c_1 \geq c_2$. Intuitively, if Player 1 can win from (F_2, c_2) then he can win from (F_1, c_1), as he has seen more accepting states and has less energy in (F_2, c_2) than in (F_1, c_1). The sets of the sequence $(W_k)_k$ are all closed for that partial order, and can thus be represented by the antichain of their maximal elements, following ideas of [14]. We also exploit this order in a forward algorithm for solving safety games as done in [14].

Incrementality Approach – The size of the parameters K and C ensuring completeness (see Theorem 9) are doubly exponential, and this is clearly impractical. Nevertheless, we can use the following property: $L_{\text{ucb},K_1}(\mathcal{A}^w, C_1) \subseteq L_{\text{ucb},K_2}(\mathcal{A}^w, C_2)$ for all $C_1 \leq C_2$ and $K_1 \leq K_2$. This inclusion tells us that if there exists a Moore machine $\mathcal{M}$ such that $L(\mathcal{M}) \subseteq L_{\text{ucb},K_1}(\mathcal{A}^w, C_1)$ then the formula is E-realizable without considering the huge theoretical bounds K and C of Theorem 9. This means that we can adopt as in [14], an incremental approach that first uses small values for parameters K and C and increments them when necessary (if the more constrained specification is not realizable).

Tool Acacia+ – In [3] we present Acacia+, a tool for LTL synthesis using antichain-based algorithms. Its main advantage, regarding other LTL synthesis tools, is to generate compact strategies that are usable in practice. This can be very useful in applications like control code synthesis from high-level LTL specifications, debugging of unrealizable LTL specifications by inspecting compact counter strategies, and generation of small deterministic Büchi or parity automata from LTL formulas (when they exist) [3].

Acacia+ is now extended to the synthesis from LTL specifications with mean-payoff objectives in the multi-dimensional setting. As explained before, it solves incrementally

Table 1. Acacia+ on the specification of Example 2 with increasing threshold values ν. The column K (resp. C) gives the minimum value (resp. vector) required to obtain a winning strategy, M the size of the finite-memory strategy, $time$ the execution time (in seconds) and mem the total memory usage (in megabytes). Note that the running time is the execution time of the forward algorithm applied to the safety game with values K and C (and not with smaller ones).

ν	-1.2	-1.02	-1.002	-1.001	-1.0002	-1.0001	-1.00005
K	4	49	499	999	4999	9999	19999
C	7	149	1499	2999	14999	29999	99999
M	5	50	500	1000	5000	10000	20000
time (s)	0.01	0.05	0.34	0.89	15.49	59.24	373
mem (MB)	9.75	9.88	11.29	12.58	30	48.89	86.68

a family of safety games, depending on some values K and C, to test whether a given specification ϕ is MP-realizable under finite memory. The tool takes as input an LTL formula ϕ with a partition of its set P of atomic signals, a weight function $w : \mathsf{Lit}(P) \mapsto \mathbb{Z}^m$, a threshold value $\nu \in \mathbb{Q}^m$, and two bounds $K \in \mathbb{Z}$ and $C \in \mathbb{Z}^m$ (the user can specify additional parameters to define the incremental policy). It then searches for a finite-memory winning strategy for Player O, within the bounds K and C, and outputs a Moore machine if such a strategy exists. The last version of Acacia+, a web interface for using it online, some benchmarks and experimental results can be found at `http://lit2.ulb.ac.be/acaciaplus/`.

Experiments – We now present some experiments. They have been done on a Linux platform with a 3.2GHz CPU (Intel Core i7) and 12GB of memory.

(1) Approaching the optimal value. Consider the specification ϕ of Example 2 and its 1-dimensional mean-payoff objective. We have shown that infinite memory strategies are required for the optimal value -1, but finite-memory ϵ-optimal strategies exist for all $\epsilon{>}0$. In Table 1, we present the experiments done for some values of $-1{-}\epsilon$.

The strategies for the system output by Acacia+ are: grant the second client $(M-1)$ times, then grant once client 1, and start over. Thus, the system almost always plays g_2w_1, except every M steps where he has to play g_1w_2. Obviously, these strategies are the smallest ones that ensure the corresponding threshold values. They can also be compactly represented by a two-state automaton with a counter that counts up to M. Let us emphasize the interest of using antichains. With $\nu = -1.001$, the underlying state space manipulated by our symbolic algorithm has a huge size: around 10^{27}, since $K = 999$, $C = 2999$ and the number of automata states is 8. However the fixpoint computed backwardly is represented by an antichain of size 2004 only.

(2) No unsollicited grants. The major drawback of the strategies presented in Table 1 is that many unsollicited grants might be sent as the strategies do not take into account client requests, and just grant the resource access to the clients in a round-robin fashion (with a longer access for client 2). It is possible to express in LTL the absence of unsollicited grants, but it is cumbersome. Alternatively, the $\mathsf{LTL}_{\mathsf{MP}}$ specification can be easily rewritten with a multi-dimensional mean-payoff objective. The specification of Examples 1 and 2 can be indeed extended with a new dimension per client, such that a request

Table 2. Acacia+ on the Shared Resource Arbiter benchmark parameterized by the number of clients, with the forward algorithm. The column c gives the number of clients, ν the threshold, K (resp. C) the minimum value (resp. vector) required to obtain a winning strategy, M the size of the finite-memory strategy, $time$ the total execution time (in seconds) and mem the total memory usage (in megabytes).

c	ν	K	C	$\lvert\mathcal{M}\rvert$	time (s)	mem (MB)
2	$(-1.2, 0, 0)$	4	$(7, 1, 1)$	11	0.02	10.04
3	$(-2.2, 0, 0, 0)$	9	$(19, 1, 1, 1)$	27	0.22	10.05
4	$(-3.2, 0, 0, 0, 0)$	14	$(12, 1, 1, 1, 1)$	65	1.52	12.18
5	$(-4.2, 0, 0, 0, 0, 0)$	19	$(29, 1, 1, 1, 1, 1)$	240	48	40.95
6	$(-5.2, 0, 0, 0, 0, 0, 0)$	24	$(17, 1, 1, 1, 1, 1, 1)$	1716	3600	636

(resp. grant) signal of client i has a reward (resp. cost) of 1 on his new dimension. More precisely, the weight function is now $w : \mathsf{Lit}(P) \to \mathbb{Z}^3$ such that $w(r_1) = (0, 1, 0)$, $w(r_2) = (0, 0, 1)$, $w(g_1) = (0, -1, 0)$, $w(g_2) = (0, 0, -2)$, $w(w_1) = (-1, 0, 0)$, $w(w_2) = (-2, 0, 0)$ and $w(l) = (0, 0, 0)$, $\forall l \in \mathsf{Lit}(P) \setminus \{r_1, r_2, g_1, g_2, w_1, w_2\}$. For $\nu = (-1, 0, 0)$, there is no hope to have a finite-memory strategy (see Example 2). For $\nu = (-1.2, 0, 0)$, Acacia+ outputs a finite-memory strategy of size 8 (with the backward algorithm) that prevents unsollicited grants. Moreover, this is the smallest strategy that ensures this threshold.

From the latter example we derive a benchmark of multi-dimensional examples parameterized by the number of clients making requests to the server. Some experimental results of Acacia+ on this benchmark are reported in Table 2.

(3) Approching the Pareto curve. As last experiment, we consider again the 2-client request-grant example with the weight function $w(w_1) = (-1, 0, 0, 0)$ and $w(w_2) = (0, -2, 0, 0)$. For this new specification there are several optimal values (w.r.t. the pairwise order), corresponding to trade-offs between the two objectives that are (i) to quickly grant client 1 and (ii) to quickly grant client 2. We try to approach, by hand, the *Pareto curve*, which consists of all those optimal values, i.e. to find finite-memory

Table 3. Acacia+ to approach Pareto values. The column ν gives the threshold, relatively close to the Pareto curve, K (resp. C) the minimum value (resp. vector) required to obtain a winning strategy, M the memory size of the strategy.

ν	K	C	M
$(-0.001, -2, 0, 0)$	999	$(1999, 1, 1, 1)$	2001
$(-0.15, -1.7, 0, 0)$	55	$(41, 55, 1, 1)$	42
$(-0.25, -1.5, 0, 0)$	3	$(7, 9, 1, 1)$	9
$(-0.5, -1, 0, 0)$	1	$(3, 3, 1, 1)$	5
$(-0.75, -0.5, 0, 0)$	3	$(9, 7, 1, 1)$	9
$(-0.85, -0.3, 0, 0)$	42	$(55, 41, 1, 1)$	9
$(-1, -0.01, 0, 0)$	199	$(1, 399, 1, 1)$	401

strategies that are incomparable w.r.t. the ensured thresholds, these thresholds being as large as possible. We give some such thresholds in Table 3, along with minimum K and C and strategy sizes. It is difficult to automatize the construction of the Pareto curve. Indeed, Acacia+ cannot test (in reasonable time) whether a formula is MP-unrealizable for a given threshold, since it has to reach the huge theoretical bound on K and C. This raises two interesting questions that we let as future work: how to decide efficiently that a formula is MP-unrealizable for a given threshold, and how to compute points of the Pareto curve efficiently.

References

1. Althoff, C.S., Thomas, W., Wallmeier, N.: Observations on determinization of Büchi automata. Theor. Comput. Sci. 363(2), 224–233 (2006)
2. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better Quality in Synthesis through Quantitative Objectives. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 140–156. Springer, Heidelberg (2009)
3. Bohy, A., Bruyère, V., Filiot, E., Jin, N., Raskin, J.-F.: Acacia+, a Tool for LTL Synthesis. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 652–657. Springer, Heidelberg (2012)
4. Boker, U., Chatterjee, K., Henzinger, T.A., Kupferman, O.: Temporal specifications with accumulative values. In: LICS, pp. 43–52. IEEE Computer Society (2011)
5. 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.) FORMATS 2008. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008)
6. Bouyer, P., Markey, N., Olschewski, J., Ummels, M.: Measuring Permissiveness in Parity Games: Mean-Payoff Parity Games Revisited. In: Bultan, T., Hsiung, P.-A. (eds.) ATVA 2011. LNCS, vol. 6996, pp. 135–149. Springer, Heidelberg (2011)
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. Chatterjee, K., Doyen, L.: Energy Parity Games. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part II. LNCS, vol. 6199, pp. 599–610. Springer, Heidelberg (2010)
9. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Log. 11(4) (2010)
10. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Generalized mean-payoff and energy games. In: FSTTCS. LIPIcs, vol. 8, pp. 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
11. Chatterjee, K., Henzinger, T.A., Jurdzinski, M.: Mean-payoff parity games. In: LICS, pp. 178–187. IEEE Computer Society (2005)
12. Chatterjee, K., Randour, M., Raskin, J.-F.: Strategy Synthesis for Multi-Dimensional Quantitative Objectives. In: Koutny, M., Ulidowski, I. (eds.) CONCUR 2012. LNCS, vol. 7454, pp. 115–131. Springer, Heidelberg (2012)
13. Ehlers, R.: Symbolic Bounded Synthesis. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 365–379. Springer, Heidelberg (2010)
14. Filiot, E., Jin, N., Raskin, J.-F.: Antichains and compositional algorithms for LTL synthesis. Formal Methods in System Design 39(3), 261–296 (2011)
15. Henzinger, T.A.: Quantitative Reactive Models. In: France, R.B., Kazmeier, J., Breu, R., Atkinson, C. (eds.) MODELS 2012. LNCS, vol. 7590, pp. 1–2. Springer, Heidelberg (2012)

16. Jobstmann, B., Bloem, R.: Optimizations for LTL synthesis. In: FMCAD, pp. 117–124. IEEE Computer Society (2006)
17. Kupferman, O., Vardi, M.Y.: Safraless decision procedures. In: FOCS, pp. 531–542. IEEE Computer Society (2005)
18. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems. Springer (1992)
19. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL, pp. 179–190. ACM Press (1989)
20. Schewe, S., Finkbeiner, B.: Bounded Synthesis. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 474–488. Springer, Heidelberg (2007)
21. Tsai, M.-H., Fogarty, S., Vardi, M.Y., Tsay, Y.-K.: State of Büchi Complementation. In: Domaratzki, M., Salomaa, K. (eds.) CIAA 2010. LNCS, vol. 6482, pp. 261–271. Springer, Heidelberg (2011)
22. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comput. Sci. 158(1&2), 343–359 (1996)