Quantitative Strategy Templates

Ashwani Anand, Satya Prakash Nayak, Ritam Raha, Irmak Sağlam, and
Anne-Kathrin Schmuck

Max Planck Institute for Software Systems, Kaiserslautern, Germany
{ashwani,sanayak,rraha,isaglam,akschmuck}@mpi-sws.org

Abstract This paper presents (permissive) *Quantitative Strategy Templates* (QaSTels) to succinctly represent infinitely many winning strategies in two-player energy and mean-payoff games. This transfers the recently introduced concept of *Permissive (qualitative) Strategy Templates* (PeSTels) for ω-regular games to games with quantitative objectives. We provide the theoretical and algorithmic foundations of (i) QaSTel synthesis, and (ii) their (incremental) combination with PeSTels for games with mixed quantitative and qualitative objectives. Using a prototype implementation of our synthesis algorithms, we demonstrate empirically that QaSTels extend the advantageous properties of strategy templates over single winning strategies – known from PeSTels – to games with (additional) quantitative objectives. This includes (i) the enhanced robustness of strategies due to their runtime-adaptability, and (ii) the compositionality of templates w.r.t. incrementally arriving objectives. We use control-inspired examples to illustrate these superior properties of QaSTels for CPS design.

1 Introduction

Two player games on finite graphs provide a powerful abstraction for modeling the strategic interactions between reactive systems and their environment. In this context, game-based abstractions are often enriched with quantitative information to model aspects like energy consumption, cost minimization, or maintaining system performance thresholds under varying conditions. As a result, games with quantitative objectives, such as energy [7] or mean-payoff [28] have gained significant attention in recent years. These games have been applied to a wide range of CPS problems, such as energy management in electric vehicles [7], optimizing resource-constrained task management in autonomous robots [18, 22], embedded systems [10], and dynamic resource allocations [5].

In practical CPS applications, strategic control decisions (i.e., the moves of the controller player in the abstract game) are typically implemented via low-level actuators. For instance, a robot's strategic decision to move to a different room involves motion control integrated with LiDAR-based obstacle avoidance. However, due to unmodeled dynamics of the physical environment that become observable only at runtime, strategic adaptations may be necessary [17, 26]. For example, if the robot detects that an entrance is obstructed by obstacles (e.g., humans), it should dynamically adjust its strategy and navigate through an alternative door instead. Therefore, synthesized (high-level) control strategies

must not only be correct-by-design, but also flexible enough to accommodate runtime adaptations. This control-inspired property of strategies has recently been formalized via permissive strategy templates (PeSTels) [2], which are similar to classical strategies but contain a vast set of relevant strategies in a succinct and simple data structure. Intuitively, PeSTels *localize* required progress towards ω-regular objectives by classifying outgoing edges of a control-player vertex as unsafe, co-live and live – indicating edges to be taken never, finitely often and infinitely often, respectively, in case the source vertex of the edge is visited infinitely often.

Inspired by PeSTels and driven by the need to capture quantitative objectives in CPS design, this paper introduces *Quantitative Strategy Templates* (QaSTels) for games with energy and mean-payoff objectives. n the context of the previously discussed robot example, such games model scenarios where a robot with limited battery must make informed re-routing decisions at runtime, ensuring that its remaining energy suffices for the required tasks. Similar to PeSTels, QaSTels localize necessary information about the *future* of the game. In contrast to PeSTels, which localize *liveness* requirements induced by a *qualitative* objective, QaSTels consider *quantitative* objectives and thereby localize the required energy loss and gain through local edge annotations. Knowing the current energy level at runtime, the control player can select from all edges that remain feasible given the available energy. This contrasts with standard game-solving approaches, which typically store only a single (optimal) action per node.

Concretely, our contributions are as follows: (i) We formalize QaSTels for energy and mean-payoff objectives, and present algorithms to extract winning strategies from them. (ii) We introduce an edge-based value iteration algorithm to compute winning QaSTels and show that QaSTels are *permissive*, i.e., they capture all winning strategies for energy objectives and all finite-memory winning strategies for mean-payoff objectives. (iii) We combine QaSTels with a bounded version of PeSTels, and propose an *efficient incremental algorithm* for updating templates and strategies under newly arriving qualitative and quantitative objectives. (iv) We highlight the advantages of strategy templates for games with quantitative objectives, and the combination of quantitative and qualitative objectives, via extensive experiments on benchmarks derived from the SYNTCOMP benchmark suit. Detailed proofs for all claims are provided in the appendix.

Related Work. The computation of permissive strategies has received significant attention over the past decade, particularly for qualitative objectives [6, 8, 20, 25]. A key development in this area is the introduction of permissive strategy templates (PeSTels) by Anand et al. [1, 2], which capture a strictly broader class of winning strategies while maintaining the same worst-case computational complexity as standard game-solving techniques. This paper extends the idea behind PeSTels to quantitative games.

When only 'classical' synthesis algorithms are available, achieving the adaptability of strategies that motivate PeSTels requires recomputing a new strategy from scratch whenever moves become unavailable at runtime or additional objec-

tives arise. For the objectives considered in this paper, this entails using 'classical' synthesis algorithms for energy objectives [7], mean-payoff objectives [9, 28], multi mean-payoff objectives [27], and mean-payoff co-Büchi objectives [11, 12]. However, since these approaches recompute strategies from scratch at each iteration, they are computationally expensive. Our benchmark experiments demonstrate that QaSTel-based adaptations offer a more efficient alternative for the applications considered.

2 Preliminaries

In this section, we introduce the basic notations used throughout the paper. We denote $\mathbb{Z}$ as the set of integers, $\mathbb{Q}$ as the set of rational numbers, $\mathbb{N}$ as the set of natural numbers including 0, and $\mathbb{N}_{>0}$ as the set of positive integers. Let $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$ and $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty, -\infty\}$. The interval $[a; b)$ represents the set $\{a, a+1, \cdots, b\}$.

2.1 Two-Player Games

A two-player game graph is a pair $G = (V, E)$, where $V = V_0 \uplus V_1$ is a finite set of nodes, $E \subseteq V \times V$ is a set of edges. The nodes are partitioned into two sets, V_0 and V_1, where V_i represents the set of nodes controlled by Player i for $i \in \{0, 1\}$. Further, we write E_i to denote the set of edges originating from nodes in V_i, i.e., $E_i = E \cap (V_i \times V)$. Given a node v, we write $E(v)$ to denote the set $\{e \in E \mid e = (v, v') \text{ for } v' \in V\}$ of all outgoing edges from v.

Value Functions. For a set of nodes V, and a set of edges E let $\mathcal{F}_V$ denote $\{f \mid f : V \to \mathbb{N}_\infty\}$, and $\mathcal{F}_E$ denote $\{f \mid f : E \to \mathbb{N}_\infty\}$.

Plays. A *play* $\rho = v_0 v_1 \ldots \in V^\omega$ on G is an infinite sequence of nodes starting from v_0 such that, $(v_i, v_{i+1}) \in E$ for each i. We denote the ith node v_i of ρ as $\rho[i]$ and use the notations $\rho[0; i] = v_0 \ldots v_i$, $\rho[i; j] = v_i \ldots v_j$, and $\rho[i; \infty] = v_i \ldots$ to denote a *prefix*, *infix*, and *suffix* of ρ, respectively. We write $v \in \rho$ (resp. $e \in \rho$) to denote that the node v (resp. the edge e) appears in ρ. Furthermore, we write $v \in \text{Inf}(\rho)$ (resp. $e \in \text{Inf}(\rho)$) to denote that node v (resp. edge e) appears infinitely often in the play ρ. We denote by $\text{plays}(G)$ the set of all plays on G, by $\text{plays}(G, v)$ denote the set of all plays starting from node v.

Strategies. A *strategy* π for Player i, where $i \in \{0, 1\}$ (or, a *Player i-strategy*) is a function $\pi : V^* \cdot V_i \mapsto E$ such that for all $H \cdot v \in V^* \cdot V_i$, we have $\pi(H \cdot v) \in E(v)$. A play $\rho = v_0 v_1 \ldots$ is called a π-play if it follows π, i.e., for all $j \in \mathbb{N}$, whenever $v_j \in V_i$ it holds that $\pi(v_0 \ldots v_j) = (v_j, v_{j+1})$. Given a strategy π, we write $\text{plays}_\pi(G, v)$ to denote the set of all π-plays starting from node v and $\text{plays}_\pi(G)$ to denote the set of all π-plays in G. For an edge e, we write $\text{plays}_\pi(G, e)$ to denote the set of plays that start with e and follows π, i.e., a play $\rho \in \text{plays}_\pi(G, e)$ iff $(\rho[0], \rho[1]) = e$ and $\rho[1; \infty] \in \text{plays}_\pi(G, \rho[1])$.

Let M be a *memory* set. A Player i-strategy π with memory M is represented as a tuple (M, m_0, α, β), where $m_0 \in M$ is the initial memory value, $\alpha : M \times$

$V \to M$ is the memory update function, and $\beta : M \times V_i \to V$ is the state transition function. Intuitively, if the current node is a Player i node v and the current memory value is m, the strategy π selects the next node $v' = \beta(m, v)$ and updates the memory to $\alpha(m, v)$. If M is finite, we call π a *finite-memory strategy*; otherwise, it is an *infinite-memory strategy*. Formally, given a history $H \cdot v \in V^* \cdot V_i$, the strategy is defined as $\pi(H \cdot v) = \beta(\hat{\alpha}(m_0, H), v)$, where $\hat{\alpha}$ is the canonical extension of α to sequences of nodes. A strategy is called *memoryless* or *positional* if $|M| = 1$. For a memoryless strategy π, it holds that $\pi(H_1 \cdot v) = \pi(H_2 \cdot v)$ for every history $H_1, H_2 \in V^*$. For convenience, we write $\pi(v)$ instead of $\pi(H \cdot v)$ for such strategies.

For a game graph $G = (V, E)$ with a finite-memory strategy $\pi = (M, m_0, \alpha, \beta)$, we denote by $G_\pi = (V', E')$ the product of G and π, that is, $V_0' = V_0 \times M$, $V_1' = V_1 \times M$, and $E' = \{((v, m), (v', m')) \mid (v, v') \in E, m' = \alpha(m, v)\}$. With slight abuse of terminology, we say that a state v is *reachable from q in* G_π if there exists a tuple (v, m') reachable from (q, m_0) in G_π. Similarly, we say that a sequence $v_0 v_1 \ldots$ is a play in G_π if there exists a corresponding play $(v_0, m_0)(v_1, m_1) \ldots$ in G_π.

Games and Objectives. A *game* is a tuple (G, φ), where G is a game graph and $\varphi \subseteq V^\omega$ is an *objective* for Player 0. A play ρ is considered *winning* if $\rho \in \varphi$. A Player 0 strategy π is *winning from a node v*, if all π-plays starting from v are winning. Similarly, π is winning from $V' \subseteq V$ if it is winning from all nodes in V'. We define the *winning region* $\texttt{Win}(G, \varphi)$ as the set of nodes from which Player 0 has a winning strategy in (G, φ). A Player 0 strategy is *winning* if it is winning from $\texttt{Win}(G, \varphi)$.

We define a *weight function* $w : E \to [-W; W]$ for some $W \in \mathbb{N}_{>0}$, which assigns an integer weight to each edge in G. This function extends naturally to finite infixes of plays, i.e., $w(v_0 v_1 \ldots v_k) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$. Furthermore, we define the average weight of a finite prefix $v_0 v_1 \ldots v_k$ as $\texttt{avg}(v_0 v_1 \ldots v_k) = \frac{1}{k} \sum_{i=0}^{k-1} w(v_i, v_{i+1})$. With this, we consider the following objectives in games:

▷ *(Quantitative) Energy Objectives.* Given a weight function w and an initial credit $c \in \mathbb{N}$, the *energy objective* is defined as $En_c(w) = \{\rho \in V^\omega \mid c + w(\rho[0; i]) \geq 0, \forall i \in \mathbb{N}\}$. Intuitively, the energy objective ensures that the total weight ('energy level') remains non-negative along a play.

▷ *(Quantitative) Mean-Payoff Objectives.* Given a weight function w, the *mean-payoff objective* is defined as[1] $MP(w) = \{\rho \in V^\omega \mid \limsup_{n \to \infty} \texttt{avg}(\rho[0; n]) \geq 0\}$. Intuitively, the mean-payoff objective ensures that the limit average weight of a play is non-negative.

▷ *(Qualitative) Parity Objectives.* Given a *priority labeling* $\mathbb{L}_P : V \to [0; d]$ for some $d \in \mathbb{N}_{>0}$, which assigns a priority to each node in G, the *parity objective* is defined as $Parity(\mathbb{L}_P) = \{\rho \in V^\omega \mid \max_{v \in \texttt{Inf}(\rho)} \mathbb{L}_P(v) \text{ is even}\}$. Intuitively, the parity objective ensures that the highest priority seen infinitely often along a play is even.

---

[1] We note that mean-payoff objectives can also be defined via the *limit-inferior* function i.e., $\{\rho \in V^\omega \mid \liminf_{n \to \infty} \texttt{avg}(\rho[0; n]) \geq 0\}$. However, it has been shown that games with either definition are equivalent [15, Corollary 8].

We refer to a game with a mean-payoff, energy, or parity objective as a *mean-payoff game*, *energy game*, and *parity game*, respectively. A game is called *mixed* if it is equipped with a conjunction of quantitative and qualitative objectives. Further, we call G weighted, if it is annotated with a weight function w, denoted by G_w. Consequently, energy and mean-payoff are referred to as weighted games.

Fixed and Unknown Initial Credit Problem. We consider the following game variants for energy objectives. **(1)** Given an initial credit c, the *energy game with fixed initial credit c* is defined as the game $(G, En_c(w))$. **(2)** A game $(G, En(w))$ with *unknown initial credit* asks Player 0 to ensure the objective $En_c(w)$ for some finite initial credit c.

In an energy game $(G = (V, E), En(w))$, there exists an optimal initial credit $\mathtt{opt}(v) \in \mathbb{Z}_\infty$ for each node v, where $\mathtt{opt}(v)$ is the minimal value (in $\mathbb{N}_\infty$) such that for every initial credit $c \geq \mathtt{opt}(v)$, there exists a winning strategy from v in the game $(G, En_c(w))$. We use $\mathtt{opt} \in \mathcal{F}_V$ to denote this *optimal value function* which assigns the optimal initial credit to each node in the game graph. It is well-known that the optimal initial credit is upper bounded by $c^* = W \cdot |V|$, where W is the maximum weight in the weight function w. Hence, for any initial credit $c \geq c^*$, the winning region for energy game $(G, En(w))$ with unknown initial credit is the same as the winning region for energy game $(G, En_c(w))$. Moreover, every winning strategy in $(G, En_c(w))$ is also winning in $(G, En(w))$.

3   Quantitative Strategy Templates (QaSTels)

In this section, we first define a quantitative strategy template (QaSTel) for weighted games and show how it can be used to represent the set of strategies in a weighted game. We then define winning and maximally permissive QaSTels.

Definition 1 (Quantitative Strategy Template (QaSTel)). *Given a game graph $G = (V, E)$, a* QaSTel *for Player 0 is a function $\Pi : V_0 \times \mathbb{N}_\infty \to 2^E$ that maps a Player 0 node u and the current credit c to a subset of outgoing edges of u in G that are* activated *by c s.t. $\Pi(u, c) \subseteq \Pi(u, c')$ for all $c' \geq c$.*

We also use a QaSTel Π as a function $V_0 \times \mathbb{Z}_\infty \to 2^E$ by extending it to negative credits as follows: $\Pi(v, c) = \emptyset$ for all $c < 0$. If $\Pi(u, i) = \Pi(u, i+1) = \cdots = \Pi(u, j) = E'$, then for notational simplicity, we will write $\Pi(u, [i; j]) = E'$. This naturally defines the *activation function* for an edge e, denoted by $act_\Pi(e)$, as the smallest value k such that $e \in \Pi(u, [k; \infty])$.

Given a weighted game G_w with a QaSTel Π and a weight $c \in \mathbb{N}$, a play $\rho = v_0 v_1 \cdots$ is said to be a (Π, c)-play if there exists a $k \in \mathbb{N}_\infty$ such that for all $i \in [0; k]$ with $v_i \in V_0$, $(v_i, v_{i+1}) \in \Pi(v_i, c + w(\rho[0; i]))$ and if $k \neq \infty$, then whenever $v_{k+1} \in V_0$ it holds that $\Pi(v_{k+1}, c + w(\rho[0; k+1])) = \emptyset$. Intuitively, either the play only uses the active edges from the QaSTel forever, or it reaches a node where no edge is active in the QaSTel and then the play continues with arbitrary edges. We collect all (Π, c)-plays in G from a node v in the set $\mathtt{plays}_\Pi(G, c, v)$. Similarly, we write $\mathtt{plays}_\Pi(G, c)$ to denote the set of all (Π, c)-plays in G.

QaSTels define a set of (Player 0) strategies in a weighted game which *follow it*, as formalized next.

Definition 2. *Given an energy game $(G, En_c(w))$ with initial credit $c \in \mathbb{N}$, a strategy π is said to* follow *a QaSTel Π, denoted by $(G, \pi) \vDash_c \Pi$ (or simply $\pi \vDash_c \Pi$ when G is clear from the context), if $\texttt{plays}_\pi(G) \subseteq \texttt{plays}_\Pi(G, c)$. Similarly, for a mean-payoff game $(G, MP(w))$, a strategy π is said to* follow *a QaSTel Π if $(G, \pi) \vDash_c \Pi$ for some $c \geq W \cdot |V|$.*

For mean-payoff games, the previous definition chooses the initial credit c to be at least the upper bound on the optimal credit, i.e., $c \geq W \cdot |V|$. This is motivated by the fact that mean-payoff games are equivalent to energy games with unknown initial credit [9]. Therefore, winning strategies of mean-payoff games can be captured by winning strategies of energy games with credit above the upper bound on the optimal credit.

In the upcoming definition, we define a *winning* QaSTel.

Definition 3 (Winning QaSTel). *Given a weighted game, a QaSTel Π is said to be* winning *from a node v (resp. a set V' of nodes) if every strategy following Π is also winning from v (resp. V'). Furthermore, a QaSTel Π is said to be* winning *if it is winning from the winning region.*

A winning QaSTel is *maximally permissive* if it includes all winning strategies.

Definition 4 (Maximal Permissiveness). *Given a weighted game, a QaSTel Π is said to be* maximally permissive *if every winning strategy follows Π. Furthermore, a QaSTel Π is said to be $\texttt{f}$-maximally permissive *if every winning strategy with finite memory follows Π.*

Given the simple and local structure of QaSTels, one can easily extract a positional strategy for Player 0 following the QaSTel by picking the edge with the smallest activation value at every node. This clearly results in a winning strategy if the QaSTel is winning.

Proposition 1. *Given a weighted game graph G_w s.t. $G = (V, E)$ with a QaSTel Π, a positional strategy π following Π can be extracted in time $\mathcal{O}(|E|)$. Let EXTRACTSTRAT(G, w, Π) be the procedure extracting this strategy.*

Proposition 2. *Given a weighted game with game graph G_w and a winning QaSTel Π, the strategy EXTRACTSTRAT(G, w, Π) is winning.*

4 Synthesizing QaSTels

We now discuss the synthesis of QaSTels over weighted games. As QaSTels are defined on edges, we first introduce an edge-optimal value function and an edge-based value iteration algorithm for weighted games. Then, we show how to extract *optimal* QaSTels from the edge-optimal value function. Finally, we show that optimal QaSTels are winning and permissive.

Edge-based Value Iteration. It is known that both energy games and mean-payoff games (with threshold 0 as considered in objective $MP(w)$) have the same value iteration algorithm [9]. To simplify the presentation we therefore restrict the discussion to energy games.

Recall that in energy games, $\mathtt{opt}(v)$ is the minimal credit required to win the game from node v. We extend this notion to $\mathtt{optE}(e)$ such that $\mathtt{optE}(e)$ is the minimal credit required to take the edge e and win the energy game from the source node of e. Formally, for some edge $e = (u, v)$, $\mathtt{optE}(e)$ is the minimal value (in $\mathbb{N}_\infty$) such that for any initial credit $c \geq \mathtt{optE}(e)$, there exists a Player 0 strategy π with $\mathtt{plays}_\pi(G, e) \subseteq En_c(w)$. To compute $\mathtt{optE}$, we extend the standard value iteration algorithm to an edge-based value iteration algorithm.

Given a weighted game graph G_w, the standard value iteration algorithm computes the least fixed point of the operator $\mathbb{O}_V : \mathcal{F}_V \to \mathcal{F}_V$ defined as:

$$\mathbb{O}_V(\mu)(u) = \begin{cases} \min\{(\mu(v) \ominus w(e)) : e = (u, v) \in E\}, \text{ if } u \in V_0 \\ \max\{(\mu(v) \ominus w(e)) : e = (u, v) \in E\}, \text{ if } u \in V_1 \end{cases} \quad (1)$$

where $l \ominus w = \max(l - w, 0)$. This fixed-point computation is initialized with an initial function $\mu_{in} : V \to \mathbb{Z}$, and each value is upper bounded by $|V| \cdot W$, i.e., once a value reaches $|V| \cdot W + 1$, we replace it by ∞. Let us denote this procedure of fixed-point computation of an operator $\mathbb{O}$ starting from an initial function μ_{in} as $\textsc{FixPoint}(G, w, \mathbb{O}, \mu_{in})$. Then, the optimal value function $\mathtt{opt}$ can be obtained by $\textsc{FixPoint}(G, w, \mathbb{O}_V, \mu_0)$ where $\mu_0(v) = 0$ for all $v \in V$.

To compute the edge-optimal value function $\mathtt{optE}$, we modify the value iteration algorithm by extending the operator $\mathbb{O}_V$ from functions over vertices to functions over edges. Hence, we define $\mathbb{O}_E : \mathcal{F}_E \to \mathcal{F}_E$ for an edge $e = (u, v)$ as:

$$\mathbb{O}_E(\mu)(e) = \begin{cases} \min\{\mu(e') \ominus w(e) : e' \in E(v)\}, \text{ if } v \in V_0 \\ \max\{\mu(e') \ominus w(e) : e' \in E(v)\}, \text{ if } v \in V_1. \end{cases} \quad (2)$$

Remark 1. It is not hard to see that the operators $\mathbb{O}_E$ and $\mathbb{O}_V$ are closely related. In particular, if $\mu_i^V \in \mathcal{F}_V$ and $\mu_i^E \in \mathcal{F}_E$ are the corresponding value functions obtained in the i-th iteration of $\mathbb{O}_V$ and $\mathbb{O}_E$ respectively, then $\mu_i^E(e) = \mu_i^V(v) \ominus w(e)$ for every edge $e = (u, v)$. This leads to a similar relation between $\mathtt{opt}$ and $\mathtt{optE}$, and hence, one can also obtain the optimal QaSTel using the standard node-based value iteration algorithm. However, our choice of presenting the edge-based approach allows us to explain our idea better, at no additional cost.

With Remark 1, the following theorem directly follows from the properties of the standard value iteration algorithm.

Theorem 1. *Given a game graph $G = (V, E)$ and weight function $w : E \to [-W, W]$, the fixed-point $\textsc{FixPoint}(G, w, \mathbb{O}_E, \mu_0)$ is the edge-optimal value function $\mathtt{optE}$ and can be computed in time $\mathcal{O}(|V| \cdot |E| \cdot W)$.*

Given a weighted game graph G_w, the winning region $\mathcal{W}$ for both mean-payoff and energy games with unknown initial credit can be extracted from the

Edge-based Value Iteration							
-	e_1 e_2 e_3 e_4 e_5 e_6 e_7						e_8
μ_0	0 0 0 0 0 0 0						0
μ_1	0 2 5 2 0 0 0						1
μ_2	0 2 5 2 0 1 0						2
μ_3	0 2 5 2 0 2 0						3
μ_4	0 2 5 2 0 3 0						4
	$\vdots$						
μ_{15}	0 2 5 2 0 14 0						15
μ_{16}	0 2 5 2 0 15 0						∞
μ_{17}	0 2 5 2 0 ∞ 0						∞

$e_1 = +1 \qquad e_5 = +1 \qquad e_8 = -1$
$e_3 = -5 \qquad e_6 = 0$

$a \qquad b \qquad c$

$e_4 = -2 \qquad e_7 = 0$
$e_2 = -2$

$$(a, [0; 2)) \mapsto \{e_1\}$$
$$(a, [2; 5)) \mapsto \{e_1, e_2\}$$
$$(a, [5; \infty)) \mapsto \{e_1, e_2, e_3\}$$
$$(b, [0; 2)) \mapsto \{e_5\}$$
$$(b, [2; \infty)) \mapsto \{e_5, e_4\}$$

Figure 1: Example of an energy game (right top) with the computation for the edge-based value iteration (left) and the optimal QaSTel (right bottom).

edge-optimal value function $\mu = \mathtt{optE}$ as

$$\mathcal{W} := \{v \in V_0 \mid \exists e \in E(v).\ \mu(e) \neq \infty\} \cup \{v \in V_1 \mid \forall e \in E(v).\ \mu(e) \neq \infty\}. \quad (3a)$$

Furthermore, for energy games with initial credit c we obtain the winning region

$$\mathcal{W}_c := \{v \in V_0 \mid \exists e \in E(v).\ \mu(e) \leq c\} \cup \{v \in V_1 \mid \forall e \in E(v).\ \mu(e) \leq c\}. \quad (3b)$$

QaSTel Extraction. Given an edge function $\mu \in \mathcal{F}_E$, we can extract a QaSTel Π from μ as follows. For every node $u \in V$ and credit $k \in \mathbb{N}_\infty$, $\Pi(u, k)$ defines the set of edges that can be taken from u with credit k, i.e.,

$$\Pi(u, k) := \{e \in E(u) \mid k \geq \mu(e)\}. \quad (4)$$

Intuitively, in an energy game, the QaSTel in (4) allows taking an edge e whenever its feasible w.r.t. edge function μ, i.e., the current energy is more than the edge value $\mu(e)$. We call the QaSTel in (4) *optimal* for the weighted game graph G_w if μ is the edge-optimal value function $\mathtt{optE}$. Given an initial edge function μ_{in}, we write COMPUTEQASTEL(G, w, μ_{in}) to denote the procedure that computes the fixed-point $\mu = \text{FIXPOINT}(G, w, \mathbb{O}_E, \mu_{in})$ and returns the corresponding winning region (as in (3)) and the corresponding QaSTel obtained from the μ (as in (4)). This means, the optimal QaSTel can be obtained by the procedure COMPUTEQASTEL(G, w, μ_0) (where μ_0 is the zero function on edges). An example of the computation of the optimal QaSTel is shown in Figure 1.

Winning and Maximally Permissive QaSTels. We now show that optimal QaSTels are winning for weighted games, and (f-)maximally permissive for (mean-payoff) energy games. As a play defined by an optimal QaSTel only takes an edge if the credit is higher than its edge-optimal value, it is winning in the energy game. Furthermore, the equivalence of energy and mean-payoff games gives the following result.

Theorem 2. *Given a weighted game graph G_w, the optimal QaSTel Π is winning in both the mean-payoff game $(G, MP(w))$ and the energy game $(G, En_c(w))$ for every initial credit $c \in \mathbb{N}$.*

As an optimal QaSTel allows every edge ensuring positive energy w.r.t. the current credit, it is maximally permissive in an energy game.

Theorem 3. *Given a weighted game graph G_w, the optimal QaSTel Π is maximally permissive in the energy game $(G, En_c(w))$ for every $c \in \mathbb{N}$.*

Unlike in energy games, the optimal QaSTels are *not* maximally permissive in mean-payoff games. However, it can capture all winning strategies with finite memory, i.e., it is f-maximally permissive. To show this, we use the following property of finite memory winning strategies in mean-payoff games.

Lemma 1. *Let $(G, MP(w))$ be a mean-payoff game with finite memory winning strategy π. Then there exists a weight bound $\mathsf{B}_\pi \in \mathbb{N}$ such that for every π-play ρ from a node $v \in \mathcal{W}(G, MP(w))$, it holds that $w(\rho[0;i]) \geq -\mathsf{B}_\pi$ for all $i \in \mathbb{N}$.*

With the above lemma, one can see that every winning strategy π in the mean-payoff game is a winning strategy in the energy game with initial credit $c = \max\{\mathsf{B}_\pi, W \cdot |V|\}$. Combining this with Theorem 3, we get the following result.

Theorem 4. *Given a weighted game graph G_w, the optimal QaSTel Π is f-maximally permissive in the mean-payoff game $(G, MP(w))$.*

This shows that a winning and permissive QaSTel can be obtained by the procedure COMPUTEQASTEL(G, w, μ_0), giving us the following result.

Corollary 1. *Given a weighted game graph G_w, a winning and maximally permissive QaSTel for the energy game $(G, En_c(w))$ can be computed in time $\mathcal{O}(|V| \cdot |E| \cdot W)$. Similarly, a winning and f-maximally permissive QaSTel for the mean-payoff game $(G, MP(w))$ can be computed in time $\mathcal{O}(|V| \cdot |E| \cdot W)$.*

5 Applications of QaSTels

As discussed in the introduction, our study of QaSTels is inspired by the advantages their qualitative counterparts – permissive strategy templates (PeSTels) for Parity games introduced in [2] – posses over classical strategies in control-inspired applications. In particular, PeSTels allow (i) to adapt winning strategies at runtime [2, 23], and (ii) to compose different templates into new ones leading to novel iterative and compositional synthesis techniques [3, 4, 24]. This section investigates whether QaSTels possess similar runtime adaptability (Section 5.1) and compositional (Section 5.2) properties.

5.1   Dynamic Strategy Extraction from QaSTels

We first consider scenarios where the runtime operation of the controlled system, e.g. a robot, is supplied with local preferences over moves that can only be determined at runtime. As an example, consider a mobile robot in a smart factory operating in the presence of (non-modeled) human operators. Here, the robot might be equipped with a perception module which predicts the probability of the successful completion of an action (e.g. reaching a certain work station) within these (dynamic) obstacles. In this case, the logical control strategy can choose the activated move from the QaSTel with the highest success probability. More generally, given a weighted game (G, φ) with optimal QaSTel Π and a dynamic preference function $\mathsf{pref}_t : E_0 \to [0, 1]$ for every $t \in \mathbb{N}_0$, we define the Player 0 strategy π_0 *online* (after obtaining pref_t in time step t) s.t.

$$\pi_0(v_0 \ldots v_t) := \arg\max\{\mathsf{pref}_t(e) \mid e \in \Pi(v_t, c_t)\}, \tag{5}$$

where c_t is the credit value at time-step t. It follows directly from the correctness of optimal QaSTels that π_0 is winning for (G, φ).

A slightly more involved scenario occurs if edges with low preference values are assumed to be blocked and hence should not be taken at all by the controlled system. This can be due to a blocking static obstacle perceived by a mobile robot, or due to an actuation failure, e.g., a faulty motor in a quad rotor, resulting in a restricted evolution of the system and hence in the unavailability of certain Player 0 moves in the game abstraction (which are henceforth assumed to be annotated with preference 0). In this case, (5) changes to

$$\pi_0(v_0 \ldots v_t) := \arg\max\{\mathsf{pref}_t(e) > \epsilon \mid e \in \Pi(v_t, c_t)\}, \tag{6}$$

for some given $\epsilon > 0$. Unfortunately, if $\pi_0(v_0 \ldots v_t)$ becomes empty, we cannot continue to control the system with the current template. However, due to the permissiveness of optimal QaSTels (see Theorems 3 and 4), such scenarios cannot occur if at least one edge with the minimal activation energy is always retained. Formally, we have the following observation.

Proposition 3. *Given a weighted game graph $(G = (V, E), w)$ with an optimal QaSTel Π, let $E_t^* := \{e \in E_0 \mid \mathsf{pref}_t(e) < \epsilon\}$ for some $t \in \mathbb{N}_0$. If there exists no $v \in V$ s.t. $\mathtt{minEdges}(v) \subseteq E_t^*$, where*

$$\mathtt{minEdges}(v) := \arg\min\{act_\Pi(e) \mid e \in E(v)\}, \tag{7}$$

then $\Pi = \textsc{computeQaSTel}(G', w, \mu_0)$ is the optimal QaSTel for $G' \setminus E_t^*$.

It follows from Proposition 3 that whenever there exists no $v \in V$ s.t. $\mathtt{minEdges}(v) \subseteq E_t^*$ for all $t \in \mathbb{N}_0$, the Player 0 strategy in (6) is winning in the original weighted game (G, φ). Furthermore, if this condition is violated at some time point t, a recomputation of QaSTels can be triggered. As an obvious corollary (see Appendix A.7) of the known monotonicity properties of the value

iteration algorithm, this recomputation can be hot-started from the current optimal value over G, i.e.,

$$\Pi' := \text{COMPUTEQASTEL}(G', w, \mu_0) = \text{COMPUTEQASTEL}(G', w, act_\Pi). \quad (8)$$

It should be noted that this dynamic recomputation of QaSTels might return an empty winning region at some time step, in which case the dynamically adapted strategy from (6) returns a finite play which is not winning anymore. Intuitively, such scenarios occur when preferences and QaSTels do not interact favorably. If preferences are due to unmodeled disturbances, such as dynamic obstacles, there is not much one can do to prevent such blocking situations. If additional objectives are, however, known and can be modelled as additional quantitative or qualitative objectives over the given game graph G, one should incorporate them into template synthesis as soon as they are available. This then leads to compositional synthesis approaches as discussed next.

5.2   Composing QaSTels

The previous section has outlined the advantages of QaSTels for the local adaptation of strategies at runtime, which is in close analogy to the properties of PeSTels [2]. Unfortunately, this section shows that QaSTels – in contrast to PeSTels – are not composable in a straightforward manner. That is, given a QaSTel Π for a weighted game graph (G, φ) and a QaSTel Π' for a different weighted game (G, φ') over the same graph, we cannot easily combine Π and Π' into a QaSTel which is winning for the combined game $(G, \varphi \wedge \varphi')$. This is due to the fact that a winning strategy for $(G, \varphi \wedge \varphi')$ might require infinite memory [27].

Nevertheless, we can still extract a single (infinite-memory) winning strategy from multiple QaSTels over mean-payoff games as long as the winning regions of all games coincide. The resulting algorithm, given in Algorithm 1, uses the function COMBINE to combine winning strategies of all games extracted from QaSTels following the procedure given in [27, Lemma 8].

Theorem 5. *Given a game graph $G = (V, E)$ with multiple* mean-payoff *objectives $\{MP(w_i)\}_{i \in [1;k]}$, COMBINEQASTEL$(G, \{MP(w_i)\}_{i \in [1;k]})$ returns a winning strategy for the game $(G, \bigwedge_{i \in [1;k]} MP(w_i))$. Furthermore, the procedure terminates in time $\mathcal{O}(k \cdot |V| \cdot |E| \cdot W)$, where W is the maximal weight in the game.*

It should be noted that COMBINEQASTEL can also be used for iterative synthesis, i.e., if a mean-payoff objective arrives, a new combined strategy can be derived by hot-starting COMBINEQASTEL.

Remark 2. We remark that games with multiple energy objectives might not always have a winning strategy, even if the winning regions of the different energy objectives coincide.

Algorithm 1 COMBINEQASTEL

Input: Game graph $G = (V, E)$ with $\{MP(w_i)\}_{i \in [1;k]}$
Output: Winning strategy π for $(G, \bigwedge_{i \in [1;k]} MP(w_i))$
 1: $\mathcal{W} = V$; $act_{\Pi_i} = \mu_0$ for all $i \in [1;k]$
 2: $(\mathcal{W}_i, \Pi_i) \leftarrow$ COMPUTEQASTEL(G, w_i, act_{Π_i})
 3: $\mathcal{W}' \leftarrow \bigcap \mathcal{W}_i$
 4: **while** $\mathcal{W} \neq \mathcal{W}'$ **do**
 5:     **for all** $e \in \mathcal{W}' \times (\mathcal{W} \setminus \mathcal{W}')$ **do** $\Pi(e) = \infty$
 6:     $(\mathcal{W}_i, \Pi_i) \leftarrow$ COMPUTEQASTEL(G, w_i, act_{Π_i})
 7:     $\mathcal{W} \leftarrow \mathcal{W}'$; $\mathcal{W}' \leftarrow \bigcap \mathcal{W}_i$
 8: $\pi_i =$ EXTRACTSTRAT(Π_i), for all $i \in [1;k]$
 9: **return** COMBINE$(\{\pi_i\}_{i \in [1;k]})$

Remark 3. It is known that parity objectives can be translated into mean-payoff objectives with threshold 0 over the same game graph in polynomial time [14, Theorem 40]. It therefore follows that the combination of QaSTels and PeSTels might require infinite memory strategies. This further implies that COMBINEQASTEL can also be used to extract combined strategies in (multi-objective) mean-payoff parity games.

6   Combining QaSTels with (bounded) PeSTels

While the previous section discussed control-inspired applications of QaSTels for runtime-adaptability and composition of templates for purely *quantitative* objectives, this section considers the construction of strategy templates for games with both *quantitative and qualitative* objectives. As Remark 3 already shows that the most general combination, i.e., mean-payoff-Parity games, require infinite strategies in general, we cannot hope for the construction of winning templates which are maximal for the full class of parity, and hence, ω-regular objectives over the weighted game graphs G_w. Instead, we propose to start with qualitative strategy templates which under-approximate the set of winning strategies for any parity objective, but allow for a straight forward combination with QaSTels, and hence for efficient incremental synthesis. These restricted qualitative templates are based on PeSTels from [2], which we recall in Section 6.1 before formalizing their composition with QaSTels in Section 6.2. Section 6.3 then discusses control-inspired mixed specifications where this class of templates ensure to capture a huge class of relevant strategies. We test the completeness and efficiency of the resulting algorithms in Section 7.

6.1   Bounded PeSTels

Within this paper we consider PeSTels which are composed of two edge conditions: (i) *unsafe* edges $S \subseteq E_0$, and (ii) *co-live* edges $D \subseteq E_0$. Their combination $\Gamma = (S, D)$ is called a *bounded* PeSTel, which represents the objective

$\mathtt{plays}_\Gamma(G) = \{\rho \in V^\omega \mid \forall e \in S : e \notin \rho \text{ and } \forall e \in D : e \notin \mathtt{Inf}(\rho)\}$. We say a strategy π (for Player 0) *follows* Γ, denoted by $(G, \pi) \models \Gamma$ (or simply $\pi \models \Gamma$ when G is clear from the context), if $\mathtt{plays}_\pi(G) \subseteq \mathtt{plays}_\Gamma(G)$. Intuitively, π follows Γ if every π-play (i) never uses the unsafe edges in S, and (ii) stops using the co-live edges in D eventually. In a qualitative game (G, Φ), a PeSTel Γ is *winning* from a node v if every strategy following Γ is also winning from v.

In [2], PeSTels include a third edge condition, called live groups, which ensures that certain edges are taken infinitely often. We discuss in Appendix A.8, how winning PeSTels for games (G, Φ) synthesized via the algorithms presented in [2] can be directly bounded leading to bounded PeSTels. We refer to the combined synthesis algorithm as COMPUTEPESTEL.

Remark 4. We note that COMPUTEPESTEL usually under-approximates the set of winning plays for (G, Φ). It is however known that for co-Büchi games no additional winning strategies can be captured by live-group templates, naturally leading to bounded templates. Notably, the algorithms for computing winning PeSTels in co-Büchi games, presented in [2], exhibit the same worst-case computation time as standard methods for solving such (finite-state) games.

6.2   Mixed Strategy Templates (MiSTels)

Following the previous discussion, this section introduces MiSTels $\Lambda = (S, D, \Pi)$ as a combination of a (bounded) PeSTel $\Gamma = (S, D)$ with a QaSTel Π defined over the same weighted game graph G_w. Thereby, MiSTels concisely represent a set of winning strategies for *mixed games* $(G, \Phi \wedge \varphi)$ which contain a quantitative objective Φ and a qualitative objective φ. A Player 0 strategy π is said to follow the MiSTel Λ over G_w if it follows both Γ and Π over G_w.

Conflict-free MiSTels. Given any combination of PeSTels and QaSTels, their direct combination might result in a MiSTel for which no strategy exists that follows it. As an example consider the discussion from Section 5.1 on the runtime adaptation of a strategy which follows a QaSTel but at the same time avoids taking unavailable edges. Given a PeSTel (S, D), the edge set S can directly be interpreted as the set of unavailable edges, which – in contrast to the case discussed in Section 5.1 – does not change and is known a priori. Similarly, the set of D collects all edges that eventually become unavailable. Following Proposition 3, we obtain the existence of a strategy following a MiSTel if $\mathtt{minEdges}(v) \not\subseteq S \cup D$ for all $v \in V$. If a MiSTel has this property, we call it *conflict free*. Similar to Proposition 1, one can extract a strategy following a conflict-free MiSTel by picking an unconstrained edge (i.e. $e \notin S \cup D$) with the smallest activation value at every node.

Proposition 4. *Given a weighted game graph G_w with a conflict-free MiSTel $\Lambda = (S, D, \Pi)$, a positional strategy following Λ can be extracted in time $\mathcal{O}(|E|)$.*

Winning MiSTels. We say that the MiSTel Λ is *winning* in the *mixed game* $(G, \Phi \wedge \varphi)$ from a node v if all strategies π that follow Λ are winning from

---

Algorithm 2 COMPUTEMISTEL(G, w, Φ)

Input: Mixed game $(G = (V, E), \varphi \wedge \Phi)$ with $\varphi = MP(w)$ or $En(w)$
Output: winning region $\mathcal{W}$, winning conflict-free MiSTel Λ

1: $act_\Pi = \mu_0$
2: $(\mathcal{W}, \mathcal{C}, \Lambda) \leftarrow$ FINDCONF(G, w, Φ, Π)
3: **while** $\mathcal{C} \neq \emptyset$ **do**
4:      $\Phi \leftarrow \Phi \wedge Safety(\mathcal{W})$
5:      **for all** $e \in \mathcal{C}$ **do** $\Pi(e) = \infty$
6:      $(\mathcal{W}, \mathcal{C}, \Lambda) \leftarrow$ FINDCONF(G, w, Φ, Π)
7: **return** $(\mathcal{W}, \Lambda)$

8: **procedure** FINDCONF(G, w, Φ, Π)
9:      $(\mathcal{W}_\Phi, (S, D)) \leftarrow$ COMPUTEPESTEL(G, Φ)
10:      $(\mathcal{W}_\varphi, \Pi) \leftarrow$ COMPUTEQASTEL(G, w, act_Π)
11:      $\mathcal{W} \leftarrow \mathcal{W}_\Phi \cap \mathcal{W}_\varphi$
12:      $\mathcal{C} = \cup_{v \in \mathcal{W}} \{ \texttt{minEdges}(v) \mid \texttt{minEdges}(v) \subseteq S \cup D \}$
13:      **return** $(\mathcal{W}, \mathcal{C}, (S, D, \Pi))$

---

v in both the quantitative game (G, φ) and the qualitative game (G, Φ). In order to synthesize a winning MiSTel for a given mixed game $(G = (V, E), \varphi \wedge \Phi)$, we can therefore iteratively construct winning PeSTels and QaSTels and remove all conflicts from their joint winning region. An efficient way to do so is formalized in Algorithm 2. After QaSTel synthesis is initialized with μ_0, winning PeSTels and QaSTels are computed (Lines 9 and 10) and conflicts in their joint winning region are detected (Line 12). If no conflict is detected, the algorithm directly terminates. Otherwise, conflicts are resolved by (i) adding an additional safety requirement to the quantitative specification Φ (Line 4), and (ii) increasing the edge-weight of conflicting edges, i.e., edges in $\texttt{minEdges}(v) \subseteq S \cup D$, to ∞ (Line 5). After this, PeSTels and QaSTels are recomputed to resolve conflicts, which might result in new ones. If no more conflicts are generated, the algorithm terminates. The resulting MiSTel is winning and conflict free, as formalized next.

Theorem 6. *Let $\mathcal{G} = (G, \varphi \wedge \Phi)$ be a mixed game with $\varphi = MP(w)$ or $En(w)$ and Φ a qualitative objective. Then, if $(\mathcal{W}, \Lambda) = COMPUTEMISTEL(G, w, \Phi)$, it holds that Λ is a conflict-free winning MiSTel from $\mathcal{W}$.*

Remark 5. We note that the iterative computation of PeSTels and QaSTels in Algorithm 2 can be hot-started, which makes COMPUTEMISTEL more efficient. For QaSTels, the correctness of hot-starting follows from Eq. (8). Notably, COMPUTEPESTEL can also be hot-started if specifications are added [2, Alg.4].

Incremental MiSTel Synthesis. Surprisingly, Algorithm 2 can directly be extended to incremental MiSTel synthesis. That is, given an already computed winning MiSTel $\Lambda = (S, D, \Pi)$ with winning region $\mathcal{W}$ for the mixed game $\mathcal{G} = (G, \varphi \wedge \Phi)$, Λ can be refined to a new winning MiSTel $\Lambda' = (S', D', \Pi')$ for the mixed game $\mathcal{G} = (G, \varphi \wedge \Phi \wedge \Phi')$ with winning region $\mathcal{W}' \subseteq \mathcal{W}$ if a new

quantitative objective Φ' arrives. For this, one would use Algorithm 2 for the combined quantitative objective $\Phi \wedge \Phi'$ and hot-starts both COMPUTEPESTEL and COMPUTEQASTEL with (S, D) and Π from the already existing MiSTel Λ.

Remark 6. We recall from Section 5.2 that adding additional *quantitative* objectives only allows to extract new winning *strategies* (and no templates) if all quantitative objectives are mean-payoff objectives. Nevertheless, this clearly can also be incorporated in an iterative version of Algorithm 2.

6.3 Applications of MiSTels

While MiSTels can be applied to any mixed game, the class of winning strategies they capture might be a quite restricted (possibly empty) subset of all available strategies. This can (i) downgrade the adaptability of strategy choices during runtime and (ii) might lead to an empty winning region in incremental or compositional synthesis approaches. MiSTels therefore have a higher potential whenever they capture a large set of winning strategies. One such example are co-Büchi games, where it is known that PeSTels are naturally bounded. After discussing how specifications commonly used in CPS applications reduce to co-Büchi games in this section, we investigate this instance further in Section 6.4.

LTL$_f$ Specifications. LTL$_f$ is a fragment of linear temporal logic (LTL) where system properties are evaluated over finite traces only, which has gained popularity in robotic applications over the last decade [13]. Specifications given in LTL$_f$ can be translated into deterministic finite automata which can be composed with a weighted game, adding a reachability objective to it. The resulting reachability objective can then be translated into a co-Büchi objective by removing all outgoing transitions from target states, adding self-loops to them and adding all states which are not a target state into the co-Büchi region.

Uniform Attractivity. If one manipulates the graph as outlined before, one can show that for every winning strategy π exists a time bound $k \in \mathbb{N}$ s.t. all plays ρ compliant with π will never leave the set of target states again after k time steps, i.e., $\forall i \geq k. \, \rho[i] \in T$ [2]. Such specifications are called uniform attractivity specifications and are used to translate classical stability objectives into formal specifications. Formally, the computation of winning strategies for general uniform attractivity games (without manipulating the game graph) are solved by first computing the winning set of the safety objective $\mathcal{W}_{safe} := Safety(T)$ and then solving the co-Büchi game $(G, co\text{-}B\ddot{u}chi(\mathcal{W}_{safe}))$. It is therefore not hard to see, that PeSTels for uniform attractivity games are also naturally bounded.

Adding Quantitative Objectives. Due to the fact that every winning strategy comes with a uniform bound on when the target set is reached, the above instances of co-Büchi games are naturally combined with qualitative energy objectives with fixed initial credit. This is in contrast to 'classical' co-Büchi objectives which are more naturally combined with mean-payoff objectives, ensuring that strategies are optimal in the limit even if they deviate for finite time durations.

---

[2] For classical co-Büchi games, such a uniform bound over all plays does in general not exist (see [16] for an example).

6.4   Mean-Payoff co-Büchi Games

Mean-payoff co-Büchi games have been already studied in [11] and will therefore be used to benchmark our prototype implementation of MiSTel synthesis against the state of the art algorithm, called `MPCoBuechi`, in Section 7. We note that these results carry over to control-inspired applications of MiSTels discussed before, as QaSTel synthesis coincides for energy and mean-payoff objectives and reachability and uniform attractivity simply utilize the (bounded) PeSTel synthesis algorithm specialized for co-Büchi games called COBÜCHITEMP from [2, Alg.2].

Complexity. Using COBÜCHITEMP from [2, Alg.2] instead of COMPUTEPESTEL in COMPUTEMISTEL (Algorithm 2), the complexity of MiSTel synthesis for mean-payoff co-Büchi games reduces to $\mathcal{O}(n^2m + nmW)$, where $n = |V|$, $m = |E|$, and W is the maximum weight in w (see Corollary 3 in Appendix A.11). In comparison, the worst-case computation time of `MPCoBuechi` is $\mathcal{O}(nmW)$ [11, Theorem 5] and, hence, lower. We however note that we achieve the same worst-case complexity when $W \geq n$, which is very often the case.

Completeness. We note that COMPUTEMISTEL is not complete. First of all, the bounded PeSTel computed by COBÜCHITEMP does not capture winning strategies for instances in which Player 1 (unexpectedly) helps Player 0 and is therefore already incomplete (see [2] for details). In addition, conflicts with co-Büchi nodes are removed immediately for conflict resolution in COMPUTEMISTEL, while they could be used by a strategy finitely often. This leads to a further potential under-approximation of the winning region, as illustrated in the example game depicted in Figure 2. Here, COBÜCHITEMP outputs the co-live edges denoted by orange dashed lines. Further, the activation energy of the edges (a, b) and (a, a) are 0 and 1 respectively, and hence, the edge (a, b) is a conflict by definition. Hence, in Algorithm 2, the edge (a, b) will be assigned the value ∞ while resolving the conflict, making the node a losing in the next iteration. However, we observe that all the nodes are winning for Player 0.

We note that the state-of-the-art algorithm by Chatterjee et al. [11] is instead complete. Our experimental results presented in Section 7 however show that the winning region computed by COMPUTEMISTEL coincides with the full winning region computed by `MPCoBuechi` for more then 90% of the considered benchmark instances.

Figure 2:   Mean-payoff   co-Büchi game with $co\text{-}B\ddot{u}chi(\{a, c\})$.

7   Empirical Evaluations

This section aims to highlight the advantages of strategy templates for games with qualitative (and quantitative) objectives. However, benchmarking QaSTel (and MiSTel) synthesis as well as their adaptability and compositional properties

(a) Fault tolerance

(b) Incremental synthesis

(c) Rounds of conflict resolution

(d) Incompleteness of `QuanTemplate`

Figure 3: Plots summarizing the experimental evaluations. Bigger figures can be found in the appendix.

is difficult for two reasons. First, to the best of our knowledge, there are no benchmark suites that include real-world applications with combined quantitative and qualitative objectives. Second, while there exist algorithms for mean-payoff parity games, we are not aware of any implementation or benchmark-based evaluation of these algorithms. To address this, we build new benchmark suites based on the SYNTCOMP benchmark [19] and implement the `MPCoBuechi` algorithm from [11] (discussed in Section 6.4) to benchmark our Java-based prototype implementation `QuanTemplate` of COMPUTEMISTEL against it.

Experimental Setup. We built a new benchmark suite from the SYNTCOMP benchmark [19] by (a) translating SYNTCOMP parity games into mean-payoff games using standard techniques [21], and (b) adding co-Büchi objectives to the qualitative games from (a) by randomly choosing avoidance regions (i.e., the set of co-Büchi nodes which a play is not allowed to visit infinitely often). For practical reasons, we imposed limits of 5×10^5 nodes and 10^5 energy values on the edges. As a result, our benchmark suite comprises 245 mean-payoff game graphs. All experiments were executed on a 32-core Debian machine equipped with an Intel Xeon E5-V2 CPU (3.3 GHz) and up to 256 GB of RAM.

Dynamic Strategy Adaptation. We have conducted experiments to evaluate the robustness of QaSTels to the unavailability of edges due to additional edge preferences used for dynamic strategy extraction at runtime. As discussed in Section 5.1, QaSTels do not need to be adapted if the edges with the minimum activation energy are still available. To evaluate how likely QaSTels need to be

recomputed we randomly removed edges from the game graphs in our benchmark suite and measured the average number of deletions required until a minimum activation edge was removed for any node. We ran `QuanTemplate` on each graph, incrementally deleting randomly selected edges until a change in the optimal value occurred. This process was repeated 10 times per graph, and we computed the average number of edge deletions needed to trigger this change. Fig. 3a illustrates the trend between the proportion of game graphs in which the optimal value changes against the proportion of Player 0 edges removed from them. The trend clearly shows that in practice, it is very unlikely that minimum activation edges are deleted even after removing a significant number of edges from the graph. This establishes the efficiency of QaSTels to produce robust strategies.

Incremental Synthesis. We have benchmarked our prototype implementation `QuanTemplate` of COMPUTEMISTEL for mean-payoff co-Büchi games against our implementation of the `MPCoBuechi` algorithm [11]. The experiment starts by providing both algorithms only with a mean-payoff game from our benchmark suite. We then add 5 co-Büchi objectives incrementally. To evaluate the dependence of runtimes on the co-Büchi objectives, we run this experiment thrice, with varying amounts of additional nodes added to the avoidance region in every incremental step: (i) 0.4% (blue circles in Fig. 3b), (ii) 6% (red triangles in Fig. 3b) and (iii) 12% (green stars in Fig. 3b) leading to (i) 2%, (ii) 30% and (iii) 60% avoidance region in the final iteration of incremental synthesis. To compare the performance of `QuanTemplate` and `MPCoBuechi`, Fig. 3b shows (i) the ratio of average runtimes of `MPCoBuechi` vs `QuanTemplate` to complete the 5 incremental synthesis steps outlined above (y-axis in Fig. 3b), against (ii) the running times of `MPCoBuechi` over the original mean-payoff game without co-Büchi objectives (x-axis in Fig. 3b).

The plot shows that when the instances are very simple (resulting in a low initial runtime of `MPCoBuechi`), `QuanTemplate` may be slow in the recomputation on additional co-Büchi objectives. However, as the graphs get more complex, `QuanTemplate` is magnitudes faster than `MPCoBuechi` in recomputing the winning regions on the fly. In fact, we see that `QuanTemplate` is around 10000 times faster on the most complex instances. This demonstrates the utility of our approach avoiding the need of recomputations from scratch.

Completeness. We conducted experiments to quantify the loss of completeness (in terms of the size of the winning region) of COMPUTEMISTEL compared to `MPCoBuechi`. For each of the 245 mean-payoff game graphs in our benchmark suite, we randomly selected (i) 25%, (ii) 50%, and (iii) 75% of the nodes and defined them as the avoidance region in the respective mean-payoff co-Büchi game. `QuanTemplate` could *not* compute the *complete* winning region in (i) 10, (ii) 30 and (iii) 28 instances of games, respectively. This shows that `QuanTemplate` computes the full winning region for over 90% of the considered games.

To further investigate the root cause of incompleteness, we evaluated the number of rounds of conflict resolutions required by `QuanTemplate` for each instance. In Fig. 3c, we plot the percentage of instances requiring a certain number of conflict resolution rounds. In Fig. 3d, we plot the percentage of incomplete

instances against the number of conflict resolution rounds required to solve the respective instance. Together the two plots present the relationship between the number of game graphs for which `QuanTemplate` was unable to synthesize a winning strategy and the number of iterations required to resolve conflicts. Fig. 3d indicates a clear trend: as the number of iterations increases, the likelihood of failure to synthesize a winning strategy also increases. However, as noted in Fig. 3c, the number of iterations required in practice remains low. This observation explains the low number of incomplete instances (less than 10%, as noted above), and supports the claim that while our algorithm may be incomplete in the worst case, it is able to synthesize winning strategies in most scenarios.

References

1. Anand, A., Mallik, K., Nayak, S.P., Schmuck, A.K.: Computing adequately permissive assumptions for synthesis. In: Tools and Algorithms for the Construction and Analysis of Systems (2023)
2. Anand, A., Nayak, S.P., Schmuck, A.K.: Synthesizing permissive winning strategy templates for parity games. In: Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part I. Lecture Notes in Computer Science (2023). https://doi.org/10.1007/978-3-031-37706-8_22
3. Anand, A., Nayak, S.P., Schmuck, A.: Strategy templates - robust certified interfaces for interacting systems. In: ATVA. Lecture Notes in Computer Science, vol. 15054, pp. 22–41. Springer (2024)
4. Anand, A., Schmuck, A.K., Prakash Nayak, S.: Contract-Based Distributed Logical Controller Synthesis. In: Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control. HSCC '24 (May 2024). https://doi.org/10.1145/3641513.3650123
5. Avni, G., Henzinger, T.A., Kupferman, O.: Dynamic resource allocation games. Theoretical Computer Science **807**, 42–55 (2020). https://doi.org/https://doi.org/10.1016/j.tcs.2019.06.031, in memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part II
6. Bernet, J., Janin, D., Walukiewicz, I.: Permissive strategies: from parity games to safety games. RAIRO Theor. Informatics Appl. **36**(3), 261–275 (2002). https://doi.org/10.1051/ita:2002013
7. 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.) Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008, Saint Malo, France, September 15-17, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5215, pp. 33–47. Springer (2008). https://doi.org/10.1007/978-3-540-85778-5_4, `https://doi.org/10.1007/978-3-540-85778-5\_4`
8. Bouyer, P., Markey, N., Olschewski, J., Ummels, M.: Measuring permissiveness in parity games: Mean-payoff parity games revisited. In: Bultan, T., Hsiung, P. (eds.) Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6996, pp. 135–149. Springer (2011). https://doi.org/10.1007/978-3-642-24372-1_11

9. Brim, L., Chaloupka, J., Doyen, L., Gentilini, R., Raskin, J.: Faster algorithms for mean-payoff games. Formal Methods Syst. Des. **38**(2), 97–118 (2011). https://doi.org/10.1007/S10703-010-0105-X, `https://doi.org/10.1007/s10703-010-0105-x`

10. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Alur, R., Lee, I. (eds.) Embedded Software, Third International Conference, EMSOFT 2003, Philadelphia, PA, USA, October 13-15, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2855, pp. 117–133. Springer (2003). https://doi.org/10.1007/978-3-540-45212-6_9, `https://doi.org/10.1007/978-3-540-45212-6\_9`

11. Chatterjee, K., Henzinger, M., Svozil, A.: Faster algorithms for mean-payoff parity games. In: Larsen, K.G., Bodlaender, H.L., Raskin, J. (eds.) 42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark. LIPIcs, vol. 83, pp. 39:1–39:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPICS.MFCS.2017.39, `https://doi.org/10.4230/LIPIcs.MFCS.2017.39`

12. Chatterjee, K., Henzinger, T.A., Jurdzinski, M.: Mean-payoff parity games. In: 20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings. pp. 178–187. IEEE Computer Society (2005). https://doi.org/10.1109/LICS.2005.26, `https://doi.org/10.1109/LICS.2005.26`

13. De Giacomo, G., Vardi, M.Y., et al.: Synthesis for ltl and ldl on finite traces. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015. pp. 1558–1564. AAAI Press (2015)

14. Fijalkow, N., Bertrand, N., Bouyer-Decitre, P., Brenguier, R., Carayol, A., Fearnley, J., Gimbert, H., Horn, F., Ibsen-Jensen, R., Markey, N., Monmege, B., Novotný, P., Randour, M., Sankur, O., Schmitz, S., Serre, O., Skomra, M.: Games on graphs. CoRR **abs/2305.10546** (2023). https://doi.org/10.48550/ARXIV.2305.10546, `https://doi.org/10.48550/arXiv.2305.10546`

15. Fijalkow, N., Bertrand, N., Bouyer-Decitre, P., Brenguier, R., Carayol, A., Fearnley, J., Gimbert, H., Horn, F., Ibsen-Jensen, R., Markey, N., Monmege, B., Novotný, P., Randour, M., Sankur, O., Schmitz, S., Serre, O., Skomra, M.: Games on graphs (2023), `https://arxiv.org/abs/2305.10546`

16. Girard, A., Eqtami, A.: Least-violating symbolic controller synthesis for safety, reachability and attractivity specifications. Automatica **127**, 109543 (2021). https://doi.org/https://doi.org/10.1016/j.automatica.2021.109543, `https://www.sciencedirect.com/science/article/pii/S0005109821000637`

17. Gittis, A., Vin, E., Fremont, D.J.: Randomized synthesis for diversity and cost constraints with control improvisation. In: CAV (2). Lecture Notes in Computer Science, vol. 13372, pp. 526–546. Springer (2022)

18. He, K., Lahijanian, M., Kavraki, L.E., Vardi, M.Y.: Reactive synthesis for finite tasks under resource constraints. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5326–5332 (2017). https://doi.org/10.1109/IROS.2017.8206426

19. Jacobs, S., Perez, G., Schlehuber-Caissier, P.: Data, scripts, and results from syntcomp 2023 (2023). https://doi.org/10.5281/zenodo.8112518

20. Klein, J., Baier, C., Klüppelholz, S.: Compositional construction of most general controllers. Acta Informatica **52**(4-5), 443–482 (2015). https://doi.org/10.1007/s00236-015-0239-9

21. Meyer, P.J., Luttenberger, M.: Solving mean-payoff games on the gpu. In: Artho, C., Legay, A., Peled, D. (eds.) Automated Technology for Verification and Analysis. pp. 262–267. Springer International Publishing, Cham (2016)

22. Muvvala, K., Lahijanian, M.: Efficient symbolic approaches for quantitative reactive synthesis with finite tasks. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 8666–8672 (2023). https://doi.org/10.1109/IROS55552.2023.10342496

23. Nayak, S.P., Egidio, L.N., Della Rossa, M., Schmuck, A.K., Jungers, R.M.: Context-triggered abstraction-based control design. IEEE Open Journal of Control Systems **2**, 277–296 (2023). https://doi.org/10.1109/OJCSYS.2023.3305835

24. Nayak, S.P., Schmuck, A.: Most general winning secure equilibria synthesis in graph games. In: Finkbeiner, B., Kovács, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 30th International Conference, TACAS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14572, pp. 173–193. Springer (2024). https://doi.org/10.1007/978-3-031-57256-2_9

25. Neider, D., Rabinovich, R., Zimmermann, M.: Down the borel hierarchy: Solving muller games via safety games. Theor. Comput. Sci. **560**, 219–234 (2014). https://doi.org/10.1016/j.tcs.2014.01.017

26. Vazquez-Chanlatte, M., Junges, S., Fremont, D.J., Seshia, S.: Entropy-guided control improvisation. In: Robotics: Science and Systems (2021)

27. Velner, Y., Chatterjee, K., Doyen, L., Henzinger, T.A., Rabinovich, A., Raskin, J.F.: The complexity of multi-mean-payoff and multi-energy games. Information and Computation **241**, 177–196 (2015). https://doi.org/https://doi.org/10.1016/j.ic.2015.03.001, `https://www.sciencedirect.com/science/article/pii/S0890540115000164`

28. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comput. Sci. **158**(1&2), 343–359 (1996). https://doi.org/10.1016/0304-3975(95)00188-3, `https://doi.org/10.1016/0304-3975(95)00188-3`

A   Missing Proofs

A.1   Proof of Proposition 1

Proposition 1. *Given a weighted game graph G_w s.t. $G = (V, E)$ with a QaSTel Π, a positional strategy π following Π can be extracted in time $\mathcal{O}(|E|)$. Let EXTRACTSTRAT(G, w, Π) be the procedure extracting this strategy.*

Proof. Consider the positional strategy π such that for every $v \in V_0$, $\pi(v)$ is an edge with the smallest activation value in Π, i.e., $\pi(v) = \arg\min\{act_\Pi(e) \mid e \in E(v)\}$. We will show that π follows Π in every weighted game with weight function w. It is enough to show that $(G, \pi) \vDash_c \Pi$ for every $c \in \mathbb{N}$.

Let $c \in \mathbb{N}$ and consider a π-play $\rho = v_0 v_1 \cdots$. For each $i \in \mathbb{N}$, let $e_i = (v_i, v_{i+1})$ and $c_i = c + w(\rho[0; i])$ be the edge and the credit after i-th step. Let $k \in \mathbb{N}_\infty$ be the smallest index such that $v_k \in V_0$ and $e_k \notin \Pi(v_k, c_k)$. Then, it is enough to show that $\Pi(v_k, c_k) = \emptyset$. Suppose $e \in \Pi(v_k, c_k)$, then $act_\Pi(e) \leq c_k$. As $\pi(v_k) = e_k$ is the edge with the smallest activation value at v_k, it holds that $act_\Pi(e_k) \leq act_\Pi(e) \leq c_k$ and hence, $e_k \in \Pi(v_k, c_k)$. This contradicts the choice of k and hence, $\Pi(v_k, c_k) = \emptyset$. $\square$

A.2   Proof of Theorem 2

Theorem 2. *Given a weighted game graph G_w, the optimal QaSTel Π is winning in both the mean-payoff game $(G, MP(w))$ and the energy game $(G, En_c(w))$ for every initial credit $c \in \mathbb{N}$.*

We will show the proof for energy games and mean-payoff games separately in the following two lemmas.

Lemma 2. *Given a game graph $G = (V, E)$ with weight function w, the optimal QaSTel Π is winning in the energy game $(G, En_c(w))$ for every initial credit $c \in \mathbb{N}$.*

Proof. Consider an energy game $(G, En_c(w))$ for some initial credit $c \in \mathbb{N}$, and let $u_0 \in V$ be a node in the winning region. We need to show that for every strategy π following Π is winning from u_0, i.e., every π-play from u_0 is winning. As every such π-play from u_0 is a (Π, c)-play from u_0, it is enough to show that every (Π, c)-play from u_0 is winning.

Let $\rho = u_0 u_1 \cdots$ be a (Π, c)-play from u_0. For each $i \in \mathbb{N}$, let us denote by $e_i = (u_i, u_{i+1})$ the edge taken at step i, and the credit at the beginning of step i by $c_i = c + w(\rho[0; i])$. By definition of Π, there exists a $k \in \mathbb{N}_\infty$ such that for all $i \in [0; k]$ with $u_i \in V_0$, $e_i \in \Pi(u_i, c_i)$, and if $k \neq \infty$, then $u_{k+1} \in V_0$ with $\Pi(u_{k+1}, c_{k+1}) = \emptyset$.

Suppose $k \neq \infty$. As $e_k \in \Pi(u_k, c_k)$, by definition of Π, $c_k \geq \mathtt{optE}(e_k)$. Hence, there exists a Player 0 strategy π such that every $\mathtt{plays}_\pi(G, e_k)$ is winning for initial credit c_k. That means, every play that starts with e_k and then follows π is winning for initial credit c_k. This implies every π-play from u_{k+1} is winning

for initial credit $c_k - w(e_k) = c_{k+1}$. Consider the strategy π' such that for every history $H \in V^* V_0$, $\pi'(H) = \pi(u_{k+1}H)$. Then, by construction, every play that starts with the edge $e'_{k+1} = \pi(u_{k+1})$ (this is well-defined as $u_{k+1} \in V_0$) and then follows π' is winning for initial credit c_{k+1}. Thus, by definition of $\mathtt{optE}$, $c_{k+1} \geq \mathtt{optE}(e'_{k+1}) = act_\Pi(e'_{k+1})$, and hence $e'_{k+1} \in \Pi(u_{k+1}, c_{k+1})$. This contradicts the assumption $\Pi(u_{k+1}, c_{k+1}) = \emptyset$. Hence, $k = \infty$.

As $k = \infty$, it holds that for every $i \in \mathbb{N}$, $e_i \in \Pi(u_i, c_i)$, and hence, by definition of QaSTels, $c_i \geq 0$. Therefore, ρ is winning. $\qquad\square$

Lemma 3. *Given a game graph $G = (V, E)$ with weight function $w : E \to [-W; W]$, the optimal QaSTel Π is winning in the mean-payoff game $(G, MP(w))$.*

Proof. Let $u_0 \in \mathcal{W}(G, MP(w))$ be a node in the winning region and let π be a strategy that follows Π in mean-payoff game $(G, MP(w))$. Then, it holds that $\pi \vDash_c \Pi$ for some $c \geq W \cdot |V|$. We need to show that every π-play from u_0 is winning.

It is well-known that the winning region of mean-payoff games is same as the winning region of energy games with unknown initial credit [9], i.e., $\mathcal{W}(G, MP(w)) = \mathcal{W}(G, En(w))$. Furthermore, as $\mathcal{W}(G, En(w)) = \mathcal{W}(G, En_c(w))$, we have $u_0 \in \mathcal{W}(G, En_c(w))$. Let ρ be a π-play from u_0. As $\pi \vDash_c \Pi$, by Lemma 2, ρ is winning in the energy game $(G, En_c(w))$. That means, for each $i \in \mathbb{N}$, $c + weight(\rho[0; i]) \geq 0$. This implies that for each $i \in \mathbb{N}$, $\mathtt{avg}(\rho[0; i]) = \frac{w(\rho[0;i])}{i} \geq \frac{-c}{i}$. Thus, $\limsup_{i \to \infty} \mathtt{avg}(\rho[0; i]) \geq \lim_{i \to \infty} \frac{-c}{i} = 0$, and hence, ρ is winning in the mean-payoff game $(G, MP(w))$. $\qquad\square$

A.3   Proof of Theorem 3

Theorem 3. *Given a weighted game graph G_w, the optimal QaSTel Π is maximally permissive in the energy game $(G, En_c(w))$ for every $c \in \mathbb{N}$.*

Proof. Consider an energy game $(G, En_c(w))$ for some initial credit $c \in \mathbb{N}$ and let π be a winning strategy. It is enough to show that for every π-play is a (Π, c)-play.

Let $\rho = u_0 u_1 \cdots$ be a π-play. For each $i \in \mathbb{N}$, let $e_i = (u_i, u_{i+1})$ be the edge taken at step i, and the credit at the beginning of step i by $c_i = c + w(\rho[0; i]) \geq 0$. For every $i \in \mathbb{N}$, let π^i be a Player 0 strategy such that for every history H, $\pi^i(H) = \pi(\rho[0; i]H)$.

Suppose ρ is a winning play, then, by construction, every π^i-play from u_{i+1} is winning for initial credit c_{i+1}. This implies that every play that starts with e_i and then follows π^i is winning for initial credit $c_{i+1} - w(e_i) = c_i$. Hence, by definition of $\mathtt{optE}$, $c_i \geq \mathtt{optE}(e_i) = act_\Pi(e_i)$, and hence, $e_i \in \Pi(u_i, c_i)$ for all $i \in \mathbb{N}$. Therefore, ρ is a (Π, c)-play.

Now, suppose ρ is not winning. As π is a winning strategy and ρ is a π-play, it holds that $u_0 \notin \mathcal{W}(G, En_c(w))$. This implies, there is no winning strategy from u_0 for initial credit c. This means, for every edge $e \in u_0 E$, there is no strategy π' such that $\mathtt{plays}_{\pi'}(G, e) \subseteq En_c(w)$, and hence, $\mathtt{optE}(e) > c$. Therefore, $\Pi(u_0, c) = \emptyset$, and hence, ρ is a (Π, c)-play. $\qquad\square$

A.4   Proof of Lemma 1

Lemma 1. *Let $(G, MP(w))$ be a mean-payoff game with finite memory winning strategy π. Then there exists a weight bound $\mathtt{B}_\pi \in \mathbb{N}$ such that for every π-play ρ from a node $v \in \mathcal{W}(G, MP(w))$, it holds that $w(\rho[0;i]) \geq -\mathtt{B}_\pi$ for all $i \in \mathbb{N}$.*

Proof. Let π be a winning strategy with finite memory in the mean-payoff game $(G, MP(w))$. Let $v \in \mathcal{W}(G, MP(w))$ be a winning node and let ρ be a π-play from v.

Consider the game graph $G_\pi = (V', E')$ induced by the strategy π. Then, every simple cycle reachable from v in the game graph G_π has a non-negative weight, the π-play from v that just loops in the cycle with negative weight has a negative limit average weight. Moreover, as ρ is a play in G_π, after an initial acyclic part of length $< |V'|$, the play ρ only visits simple cycles of G_π. As all such cycles have non-negative weight, the total weight of every prefix of ρ must be at least $-2\,|V'| \cdot W$, i.e., $w(\rho[0;i]) \geq -2\,|V'| \cdot W$ for all $i \in \mathbb{N}$. As v_0 and ρ are arbitrary, for weight bound $\mathtt{B}_\pi = 2\,|V'| \cdot W$, the lemma holds.   □

A.5   Proof of Theorem 4

Theorem 4. *Given a weighted game graph G_w, the optimal QaSTel Π is $\mathtt{f}$-maximally permissive in the mean-payoff game $(G, MP(w))$.*

Proof. Let π be a winning strategy with finite memory in the mean-payoff game $(G, MP(w))$. We need to show that every $\pi \vDash_c \Pi$ for some $c \geq W \cdot |V|$, where W is the maximum weight in the game.

By Lemma 1, there exists a weight bound $\mathtt{B}_\pi \in \mathbb{N}$ such that for every π-play ρ from a winning node $v \in \mathcal{W}(G, MP(w))$, it holds that $w(\rho[0;i]) \geq -\mathtt{B}_\pi$ for all $i \in \mathbb{N}$. Let $c = \max\{W \cdot |V|, \mathtt{B}_\pi\}$. Since $\mathcal{W}(G, MP(w)) = \mathcal{W}(G, En_c(w))$ (as discussed in the proof of Lemma 3), it holds that for every π-play ρ from a node $v \in \mathcal{W}(G, En_c(w))$, we have $c + w(\rho[0;i]) \geq \mathtt{B}_\pi + w(\rho[0;i]) \geq 0$ for all $i \in \mathbb{N}$. Hence, by definition, π is a winning strategy in the energy game $(G, En_c(w))$. Then, by Theorem 3, $\pi \vDash_c \Pi$.   □

A.6   Proof of Theorem 5

Theorem 5. *Given a game graph $G = (V, E)$ with multiple mean-payoff objectives $\{MP(w_i)\}_{i \in [1;k]}$, $\mathrm{COMBINEQASTEL}(G, \{MP(w_i)\}_{i \in [1;k]})$ returns a winning strategy for the game $(G, \bigwedge_{i \in [1;k]} MP(w_i))$. Furthermore, the procedure terminates in time $\mathcal{O}(k \cdot |V| \cdot |E| \cdot W)$, where W is the maximal weight in the game.*

Proof. Let $\mathcal{W}^j$ and $\mathcal{W}_i^j$ be the corresponding region, and Π_i^j be the corresponding QaSTel in the j-th iteration of the while loop. As $\mathcal{W}_i^j$ is a winning region computed by COMPUTEQASTEL, there is no Player 1 edge from $\mathcal{W}_i^j$ to outside of $\mathcal{W}_i^j$. Hence, all the edges from W^j to outside of $\mathcal{W}^j$ are Player 0 edges. Furthermore, since in every iteration, we are hot-starting the procedure COMPUTEQASTEL

from previous iteration by making activation values of some of these Player 0 edges ∞, the winning region $\mathcal{W}^{j+1}$ is a subset of W^j. Hence, the while loop terminates within $|V|$ iterations. As the algorithm only hot-starts the value iteration algorithm for each objective, the time complexity of the algorithm is $O(k \cdot |V| \cdot |E|)$.

Now, suppose k is the last iteration. Then, as Π_i^k is obtained from COMPUTEQASTEL$(G, w_i, \cdot)$, it holds that Π_i^k is winning from $\mathcal{W}^k$ in the mean-payoff game $(G, MP(w_i))$. Hence, every strategy π_i obtained in the algorithm after termination of while loop is a winning strategy from $\mathcal{W}^k$ in the mean-payoff game $(G, MP(w_i))$. Then, by the property of COMBINE, COMBINE$(\pi_1, \ldots, \pi_k)$ is a winning strategy from $\mathcal{W}^k$ in the mean-payoff game $(G, \bigwedge_{i \in [1;k]} MP(w_i))$. Hence, it is enough to show that $\mathcal{W}^k$ is the winning region $\mathcal{W}'$ of the mean-payoff game $(G, \bigwedge_{i \in [1;k]} MP(w_i))$.

By the previous argument, $\mathcal{W}^k \subseteq \mathcal{W}'$. Now, let us show using induction on $j \in [1;k]$ that $\mathcal{W}^j \supseteq \mathcal{W}'$. For base case $j = 1$, as intersection of winning regions of each mean-payoff objective contains the winning region of their conjunction, it holds that $\mathcal{W}^1 \supseteq \mathcal{W}'$. Now, suppose $\mathcal{W}^j \supseteq \mathcal{W}'$ for some $j \in [1; k-1]$. Since removing edges going out of winning region does not change the winning region, making activation values of some of these edges ∞ does not change the winning region. Hence, $\mathcal{W}^{j+1} \supseteq \mathcal{W}'$. Therefore, $\mathcal{W}^k = \mathcal{W}'$. $\square$

A.7   Proof of Corollary 2

Corollary 2. *Given the premises of Prop. 3, it holds that the QaSTel $\Pi' := $ COMPUTEQASTEL$(G', w, \mu_0) = $ COMPUTEQASTEL(G', w, act_Π) is optimal for G'.*

Proof. Let the operator for edge-based value iteration for game $(G', En(w))$ be $\mathbb{O}'_E$, and let optE and optE$'$ be the edge-optimal value functions for $(G, En(w))$ and $(G', En(w))$ respectively. As the activation function of the optimal QaSTel is same as the edge-optimal value function, it is enough to show that the fixed point computation of $\mathbb{O}'_E$ starting from optE gives the function optE$'$.

Let us first shows that optE$(e) \leq$ optE$'(e)$ for all $e \in E \setminus \{e_*\}$. Let $e \in E \setminus \{e_*\}$ be an arbitrary edge. If optE$'(e) = \infty$, then optE$(e) \leq$ optE$'(e)$ trivially holds. Suppose optE$'(e) \neq \infty$, then by definition, for every initial credit $c \geq$ optE$'(e)$, there exists a Player 0 strategy π_c with plays$(G', \pi_c) \subseteq En_c(w)$. As $e_* \in E_0$, the strategy π_c is also a well-defined Player 0 strategy in game graph G. Hence, for every initial credit $c \geq$ optE$'(e)$, we have plays$(G, \pi_c) \subseteq En_c(w)$, which implies optE$(e) \geq$ optE$'(e)$.

As optE$'$ is the least fixed point of the monotonic operator $\mathbb{O}'_E$ and optE $\leq$ optE$'$, by the Knaster-Tarski theorem, the fixed point computation of $\mathbb{O}'_E$ starting from optE gives the function optE$'$. $\square$

A.8   Bounding PeSTels

Given a game graph $G = (V, E)$, PeSTels (as defined in [2]) contain three types of edge templates: (i) *unsafe* edges $S \subseteq E_0$, (ii) *co-live* edges $D \subseteq E_0$ and (iii)

live groups $H_\ell \subseteq 2_0^E$. Their combination $\Gamma = (S, D, H_\ell)$ is called a PeSTel, which represents the objective $\texttt{plays}_\Gamma(G) = \{\rho \in V^\omega \mid \forall e \in S : e \notin \rho \text{ and } \forall e \in D : e \notin \texttt{Inf}(\rho) \text{ and } \forall H \in H_\ell : src(H) \cap \texttt{Inf}(\rho) \neq \emptyset \Rightarrow H \cap \texttt{Inf}(\rho) \neq \emptyset\}$, where $src(H)$ denotes the source nodes of the edges in H.

Within this paper, we restrict our attention to PeSTels which only consist of safety & co-live templates and call such PeSTels *bounded* as constraint edges can never be taken unboundedly.

Given a quantitative objective Φ which can be modelled as a parity objective over G, one can compute a bounded PeSTel $\Gamma = (S, D)$ for (G, Φ) by using the algorithm PARITYTEMPLATE from [2, Alg.3] to compute a (full) PeSTel $\widetilde{\Gamma} = (S, D, H_\ell)$ first. This PeSTel $\widetilde{\Gamma}$ can then be *bounded* into a bounded template Γ by adding all non-live outgoing edges of a live-group node, i.e. $\{(q, q') \notin H \cup S \mid (q, \cdot) \in H\}$ to D, and then setting $H_\ell := \emptyset$.

A.9   Proof of Proposition 4

Proposition 4. *Given a weighted game graph G_w with a conflict-free MiSTel $\Lambda = (S, D, \Pi)$, a positional strategy following Λ can be extracted in time $\mathcal{O}(|E|)$.*

Proof. By the definition of quantitative conflict-freeness, for every node $v \in V_0$, there is an edge $e \in \texttt{minEdges}(v)$ that is neither colive nor unsafe. With this, let us define a positional strategy π such that for every node $v \in V_0$, $\pi(v)$ is an edge $e \in \texttt{minEdges}(v)$ that is neither colive nor unsafe. Then, by the definition colive edges, π follows the PeSTel (S, D). Furthermore, by the proof of Proposition 1, π also follows the QaSTel Π. Therefore, π follows the mixed template Λ. $\square$

A.10   Proof of Theorem 6

Theorem 6. *Let $\mathcal{G} = (G, \varphi \wedge \Phi)$ be a mixed game with $\varphi = MP(w)$ or $En(w)$ and Φ a qualitative objective. Then, if $(\mathcal{W}, \Lambda) = COMPUTEMISTEL(G, w, \Phi)$, it holds that Λ is a conflict-free winning MiSTel from $\mathcal{W}$.*

Proof. Let $\Lambda = (S, D, \Pi)$. The template Λ is trivially conflict-free due to the while loop in the algorithm. Let after i-th iteration of the while loop, Φ^i be the qualitative objective, (S^i, D^i) be the PeSTel obtained, and Π^i be the QaSTel obtained. Moreover, let $\mathcal{W}_\Phi^i$ and $\mathcal{W}_\varphi^i$ be the corresponding regions after the i-th iteration. For soundness, we need to show that for every $i \geq 0$, Λ^i is a winning mixed template from $\mathcal{W}^i = \mathcal{W}_\Phi^i \cap \mathcal{W}_\varphi^i$ in game $\mathcal{G}$.

Fix some $i \geq 0$. Let $u \in \mathcal{W}^i$ and let π be a strategy following Λ^i, i.e., $(G, \pi) \vDash_c \Pi^i$ for some $c \geq W \cdot |V|$ and $(G, \pi) \vDash (S^i, D^i)$. By soundness of COMPUTEPESTEL, π is winning from u for objective Φ^i. As in each iteration, we are just adding additional safety objectives to Φ, π is also winning from u for objective Φ. Furthermore, as PeSTels only marks Player 0's edges as unsafe or co-live, the conflict edges only belong to Player 0. Hence, by Corollary 2, π is winning from u for objective φ. So, π is a strategy that does not use conflict edges and is winning from u for objective φ. In total, π is winning from u for both objectives Φ and φ, and hence, is winning from u in game $\mathcal{G}$. $\square$

A.11   Proof of Corollary 3

Corollary 3. *Given a mean-payoff co-Büchi game $(G, MP(w) \wedge co\text{-}B\ddot{u}chi(T))$, if $(\mathcal{W}, \Lambda) = \textsc{computeMiStel}(G, w, co\text{-}B\ddot{u}chi(T))$, then Λ is a conflict-free winning mixed template from $\mathcal{W}$. Furthermore, the procedure terminates in time $\mathcal{O}(n^2m + nmW)$, where $n = |V|$, $m = |E|$, and W is the maximum weight in w.*

Proof. As the soundness directly follows from Theorem 6, we only need to show the complexity. We know that each round of while loop uses at most one call to coBüchiTemp and one call to computeQaStel. Furthermore, if the winning region does not change in an iteration, then coBüchiTemp need not be called as the qualitative objective remains the same. Then coBüchiTemp will be called at most n times, which takes time $\mathcal{O}(n^2m)$ in total. Moreover, since we are hot-starting the computeQaStel algorithm, in total, the time taken by computeQaStel across all iterations is $\mathcal{O}(nmW)$. Hence, the algorithm terminates in time $\mathcal{O}(n^2m + nmW)$.   □

B   Plots from the experiments

(a) Fault tolerance

(b) Incremental synthesis

(d) Incompleteness of `QuanTemplate`

Figure 4: Plots summarizing the experimental evaluations.