On Satisficing in Quantitative Games

Abstract. Several problems in planning and reactive synthesis can be reduced to the analysis of two-player quantitative graph games. *Optimization* is one form of analysis. We argue that in many cases it may be better to replace the optimization problem with the *satisficing problem*, where instead of searching for optimal solutions, the goal is to search for solutions that adhere to a given threshold bound.
This work defines and investigates the satisficing problem on a two-player graph game with the discounted-sum cost model. We show that while the satisficing problem can be solved using numerical methods just like the optimization problem, this solution does not render compelling benefits over optimization. When the discount factor is, however, an integer, we present another solution to satisficing, which is purely based on automata methods. We show that this solution is algorithmically more performant - both theoretically and empirically - and demonstrates the broader applicability of the satisficing over optimization.

1 Introduction

Quantitative properties of systems are increasingly being explored in automated reasoning [4,14,15,19,20,24]. In decision-making domains such as planning and reactive synthesis, quantitative properties have been deployed to describe *soft constraints* such as quality measures [11], cost and resources [17,21], rewards [29], and the like. Since these constraints are soft, it suffices to generate solutions that are *good enough* w.r.t. the quantitative property.

Existing approaches on the analysis of quantitative properties have, however, primarily focused on *optimization* of these constraints, i.e., to generate optimal solutions. We argue that there may be disadvantages to searching for optimal solutions, where *good enough* ones may suffice. First, optimization may be more expensive than searching for good-enough solutions. Second, optimization restricts the search space of possible solutions, and thus could limit the broader applicability of the resulting solutions. For instance, to generate solutions that operate *within* battery life, it is too restrictive to search for solutions with *minimal* battery consumption. Besides, solutions with minimal battery consumption may be limited in their applicability, since they may not satisfy other goals, such as desirable temporal tasks.

To this end, this work focuses on directly searching for good-enough solutions. We propose an alternate form of analysis of quantitative properties in which the objective is to search for a solution that adheres to *a given threshold constraint*. We call this the *satisficing problem*, a term popularized by H.A.Simon in economics to mean *satisfy and suffice*, implying a search for good-enough solutions [1]. Through theoretical and empirical investigation, we make the case

that the satisficing is algorithmically more performant than optimization and, further, that satisficing solutions have broader applicability than optimal solutions.

This work formulates and investigates the satisficing problem on two-player, finite-state games with the discounted-sum (DS) aggregate function as the cost model, which is a standard cost-model in decision making domains [22,23,26]. In these games, players take turns to pass a token along the *transition relation* between the states. As the token is pushed around, the play accumulates costs along the transitions using the DS cost model. The players are assumed to have opposing objectives: one player maximizes the cost, while the other player minimizes it. We define the satisficing problem as follows: *Given a* threshold value $v \in \mathbb{Q}$*, does there exist a strategy for the minimizing (or maximizing) player that ensures the cost of all resulting plays is strictly or non-strictly lower (or greater) than the threshold* v*?*

Clearly, the satisficing problem is decidable since the optimization problem on these quantitative games is known to be solvable in pseudo-polynomial time [16,30]. To design an algorithm for satisficing, we first adapt the celebrated value-iteration (VI) based algorithm for optimization [30]. We show, however, that this algorithm, called VISatisfice, displays the same complexity as optimization and hence renders no complexity-theoretic advantage. To obtain worst-case complexity, we perform a thorough worst-case analysis of VI for optimization. It is interesting that a thorough analysis of VI for optimization had hitherto been absent from the literature, despite the popularity of VI. To address this gap, we first prove that VI should be executed for $\Theta(|V|^2)$ iterations to compute the optimal value, where V and E refer to the sets of states and transitions in the quantitative game. Next, to compute the overall complexity, we take into account the cost of arithmetic operations as well, since they appear in abundance in VI. We demonstrate an orders-of-magnitude difference between the complexity of VI under different cost-models of arithmetic. For instance, for integer discount factors, we show that VI is $\mathcal{O}(|V|^2) \cdot |E|)$ and $\mathcal{O}(|V|^4 \cdot |E|)$ under the unit-cost and bit-cost models of arithmetic, respectively. Clearly, this shows that VI for optimization, and hence VISatisfice, does not scale to large quantitative games.

We then present a purely automata-based approach for satisficing. While this approach applies to integer discount factors only, this approach solves satisficing in $\mathcal{O}(|V| + |E|)$ time. This shows that there is a fundamental separation in complexity between satisficing and VI-based optimization, as even the lower bound on the number of iterations in VI is higher. In this approach, the satisficing problem is reduced to solving a safety or reachability game. Our core observation is that the criteria to fulfil satisficing with respect to threshold value $v \in \mathbb{Q}$ can be expressed as membership in an automaton that accepts a weight sequence A iff $DS(A, d)$ R v holds, where $d > 1$ is the discount factor and $\mathsf{R} \in \{\leq, \geq, <, >\}$. In existing literature, such automata are called *comparator automata* (comparators, in short) when the threshold value $v = 0$ [6,7]. They are known to have a compact safety or co-safety automaton representation [9,18], which could be used to reduce the satisficing problem with zero threshold value. To solve satisficing

for arbitrary threshold values $v \in Q$, we extend existing results on comparators to permit arbitrary but fixed threshold values $v \in \mathbb{Q}$. An empirical comparison between the performance of VISatisfice, VI for optimization, and automata-based solution for satisficing shows that the latter outperforms the others in runtime and number of benchmarks, as per the complexity analysis.

In addition to improved algorithmic performance, we establish that satisficing solutions have broader applicability than optimal ones. We examine this with respect to their ability to extend to temporal goals. That is, the problem is to find optimal/satisficing solutions that also satisfy a given temporal goal. Prior results have shown this to not be possible with optimal solutions [13]. In contrast, we show satisficing extends to temporal goals when the discount factor is an integer. This occurs because both satisficing and satisfaction of temporal goals are solved via automata-based techniques, which can be easily integrated.

In summary, this work contributes to showing that satisficing has algorithmic and applicability advantages over optimization in (deterministic) quantitative games. In particular, we have shown that the automata-based approach for satisficing is of benefit over approaches like VI that are based on numerical methods. This gives yet another evidence in favor of automata-based quantitative reasoning and opens up several compelling directions for future work.

A long version of this paper with complete proofs can be found [**?**].

2 Preliminaries

2.1 Two-player graph games

Reachability and safety games. Both *reachability* and *safety games* are defined over the structure $G = (V = V_0 \uplus V_1, v_{\mathsf{init}}, E, \mathcal{F})$ [28]. It consists of a directed graph (V, E), and a partition (V_0, V_1) of its states V. State v_{init} is the *initial state* of the game. vE designates the set of successors of state v. For convenience, we assume that every state has at least one outgoing edge, i.e, $vE \neq \emptyset$ for all $v \in V$. $\mathcal{F} \subseteq V$ is a non-empty set of states. $\mathcal{F}$ is referred to as *accepting* and *rejecting* states in reachability and safety games, respectively.

A *play* of a game involves two players, denoted by P_0 and P_1, to create an infinite path by moving a token along the transitions as follows: At the beginning, the token is at the initial state. If the current position v belongs to V_i, then P_i chooses the successor state from vE. Formally, a play $\rho = v_0 v_1 v_2 \dots$ is an infinite sequence of states such that the first state $v_0 = v_{\mathsf{init}}$, and each pair of successive states is a transition, i.e., $(v_k, v_{k+1}) \in E$ for all $k \geq 0$. A play is *winning for player* P_1 in a reachability game if it visits an accepting state, and *winning for player* P_0 otherwise. The opposite holds in safety games, i.e., a play is winning for player P_1 if it does not visit any rejecting state, and winning for P_0 otherwise.

A *strategy* for a player is a recipe that guides the player on which state to go next to based on the history of the play. A *strategy is winning for a player* P_i if for all strategies of the opponent player P_{1-i}, the resulting plays are winning for P_i. To *solve* a graph game means to determine whether there exists a winning strategy for player P_1. Reachability and safety games are solved in $\mathcal{O}(|V| + |E|)$.

Quantitative graph games. A *quantitative graph game* (or quantitative game, in short) is defined over a structure $G = (V = V_0 \uplus V_1, v_{\mathsf{init}}, E, \gamma)$. V, V_0, V_1, v_{init}, E, plays and strategies are defined as earlier. Each transition of the game is associated with a *cost* determined by the *cost function* $\gamma : E \to \mathbb{Z}$. The *cost sequence* of a play ρ is the sequence of costs $w_0 w_1 w_2 \ldots$ such that $w_k = \gamma((v_k, v_{k+1}))$ for all $i \geq 0$. Given a discount factor $d > 1$, the *cost of play* ρ, denoted $wt(\rho)$, is the discounted sum of its cost sequence, i.e., $wt(\rho) = DS(\rho, d) = w_0 + \frac{w_1}{d} + \frac{w_2}{d^2} + \ldots$.

2.2 Automata and formal languages

Büchi automata. A *Büchi automaton* is a tuple $\mathcal{A} = (S, \Sigma, \delta, s_{\mathcal{I}}, \mathcal{F})$, where S is a finite set of *states*, Σ is a finite *input alphabet*, $\delta \subseteq (S \times \Sigma \times S)$ is the *transition relation*, state $s_{\mathcal{I}} \in S$ is the *initial state*, and $\mathcal{F} \subseteq S$ is the set of *accepting states* [28]. A Büchi automaton is *deterministic* if for all states s and inputs a, $|\{s' | (s, a, s') \in \delta \text{ for some } s'\}| \leq 1$. For a word $w = w_0 w_1 \cdots \in \Sigma^\omega$, a *run* ρ of w is a sequence of states $s_0 s_1 \ldots$ s.t. $s_0 = s_{\mathcal{I}}$, and $\tau_i = (s_i, w_i, s_{i+1}) \in \delta$ for all i. Let $inf(\rho)$ denote the set of states that occur infinitely often in run ρ. A run ρ is an *accepting run* if $inf(\rho) \cap \mathcal{F} \neq \emptyset$. A word w is an accepting word if it has an accepting run. The language of Büchi automaton $\mathcal{A}$ is the set of all words accepted by $\mathcal{A}$. Languages accepted by Büchi automata are called *ω-regular*.

Safety and co-safety languages. Let $\mathcal{L} \subseteq \Sigma^\omega$ be a language over alphabet Σ. A finite word $w \in \Sigma^*$ is a *bad prefix* for $\mathcal{L}$ if for all infinite words $y \in \Sigma^\omega$, $x \cdot y \notin \mathcal{L}$. A language $\mathcal{L}$ is a *safety language* if every word $w \notin \mathcal{L}$ has a bad prefix for $\mathcal{L}$ [3]. A *co-safety language* is the complement of a safety language [18]. Safety and co-safety languages that are ω-regular are represented by specialized Büchi automata called *safety* and *co-safety automata*, respectively.

Comparison language and comparator automata. Given integer bound $\mu > 0$, discount factor $d > 1$, and relation $\mathsf{R} \in \{<, >, \leq, \geq, =, \neq\}$ the *comparison language with upper bound μ, relation* R*, discount factor d* is the language of words over the alphabet $\Sigma = \{-\mu, \ldots, \mu\}$ that accepts $A \in \Sigma^\omega$ iff $DS(A, d)\ \mathsf{R}\ 0$ holds [5,9]. The *comparator automata with upper bound μ, relation* R*, discount factor d* is the automaton that accepts the corresponding comparison language [6]. Depending on R, these languages are safety or co-safety [9]. A comparison language is said to be *ω-regular* if its automaton is a Büchi automaton. Comparison languages are ω-regular iff the discount factor is an integer [7].

3 Satisficing via Optimization

This section shows that there are no complexity-theoretic benefits to solving the satisficing problem via algorithms for the optimization problem.

We begin this section by reviewing the celebrated value-iteration (VI) algorithm for optimization by Zwick and Patterson (ZP), and formally defining the

satisficing problem in § 3.1. While ZP *claim without proof* that the algorithm runs in pseudo-polynomial time [30], its worst-case analysis is absent from literature. This section presents a detailed account of the said analysis, and exposes the dependence of VI's worst-case complexity on the discount factor $d > 1$ and the cost-model for arithmetic operations i.e. unit-cost or bit-cost model. The analysis is split into two parts: First, in § 3.2 we show that it is sufficient to terminate after a finite-number of iterations. Next, we account for the cost of arithmetic operations per iteration to compute the algorithm's worst-case complexity under unit- and bit-cost cost models of arithmetic § 3.3. Finally, our VI-based algorithm for satisficing VISatisfice is presented and analyzed in § 3.4.

3.1 Satisficing and Optimization

Definition 1 (Optimization problem). *Given a quantitative graph game G, the* optimization problem *is to compute the optimal cost from all possible plays from the game, under the assumption that the players have opposing objectives to maximize and minimize the cost of plays, respectively.*

Seminal work by Zwick and Patterson showed the optimization problem is solved by the value-iteration algorithm presented here [30]. Essentially, the algorithm plays a min-max game between the two players. Let $wt_k(v)$ denote the optimal cost of a k-length game that begins in state $v \in V$. Then $wt_k(v)$ can be computed using the following equations: The optimal cost of a 1-length game beginning in state $v \in V$ is $\max\{\gamma(v,w)|(v,w) \in E\}$ if $v \in V_0$ and $\min\{\gamma(v,w)|(v,w) \in E\}$ if $v \in V_1$. Given the optimal-cost of a k-length game, the optimal cost of a $(k+1)$-length game is computed as follows:

$$wt_{k+1}(v) = \begin{cases} max\{\gamma(v,w) + \frac{1}{d} \cdot wt_k(w)|(v,w) \in E\} \text{ if } v \in V_0 \\ min\{\gamma(v,w) + \frac{1}{d} \cdot wt_k(w)|(v,w) \in E\} \text{ if } v \in V_1 \end{cases}$$

Let W be the optimal cost. Then, $W = \lim_{k\to\infty} wt_k(v_{\mathsf{init}})$. [25,30].

Definition 2 (Satisficing problem). *Given a quantitative graph game G and a threshold value $v \in \mathbb{Q}$, the* satisficing problem *is to determine whether the minimizing (or maximizing) player has a strategy that ensures the cost of all resulting plays is strictly or non-strictly lower (or greater) than the threshold v.*

3.2 Number of iterations

The VI algorithm described above terminates at *infinitum*. To compute the algorithms' worst-case complexity, we establish a crisp bound on the number of iterations that is sufficient to compute the optimal cost. We also establish a matching lower bound, showing that our analysis is tight.

Upper bound on number of iterations. The upper bound computation utilizes one key result from existing literature: There exist memoryless strategies for both players such that the cost of the resulting play is the optimal cost [25]. Then, there must exists an optimal play in the form of a *simple lasso* in the quantitative game, where a *lasso* is a play represented as $v_0v_1 \dots v_n(s_0s_2 \dots s_m)^\omega$. We call the initial segment $v_0v_1 \dots v_n$ its *head*, and the cycle segment $s_0s_1 \dots s_m$ its *loop*. A lasso is *simple* if each state in $\{v_0 \dots v_n, s_0, \dots s_m\}$ is distinct. We begin our proof by assigning constraints on the optimal cost using the simple lasso structure of an optimal play (Corollary 1 and Corollary 2).

Let $l = a_0 \dots a_n(b_0 \dots b_m)^\omega$ be the cost sequence of a lasso such that $l_1 = a_0 \dots a_n$ and $l_2 = b_0 \dots b_m$ are the cost sequences of the head and the loop, respectively. Then the following can be said about $DS(l_1 \cdot l_2^\omega, d)$,

Lemma 1. *Let $l = l_1 \cdot (l_2)^\omega$ represent an integer cost sequence of a lasso, where l_1 and l_2 are the cost sequences of the head and loop of the lasso. Let $d = \frac{p}{q}$ be the discount factor. Then, $DS(l, d)$ is a rational number with denominator at most $(p^{|l_2|} - q^{|l_2|}) \cdot (p^{|l_1|})$.*

Lemma 1 is proven by unrolling $DS(l_1 \cdot l_2^\omega, d)$. The full proof is in the supplemental material. Then, the first constraint on the optimal cost is as follows:

Corollary 1. *Let $G = (V, v_{\mathsf{init}}, E, \gamma)$ be a quantitative graph game. Let $d = \frac{p}{q}$ be the discount factor. Then the optimal cost of the game is a rational number with denominator at most $(p^{|V|} - q^{|V|}) \cdot (p^{|V|})$*

Proof. Recall, there exists a simple lasso that computes the optimal cost. Since a simple lasso is of $|V|$-length at most, the length of its head and loop are at most $|V|$ each. So, the expression from Lemma 1 simplifies to $(p^{|V|} - q^{|V|}) \cdot (p^{|V|})$. □

The second constraint has to do with the minimum non-zero difference between the cost of simple lassos:

Corollary 2. *Let $G = (V, v_{\mathsf{init}}, E, \gamma)$ be a quantitative graph game. Let $d = \frac{p}{q}$ be the discount factor. Then the minimal non-zero difference between the cost of simple lassos is a rational with denominator at most $(p^{(|V|^2)} - q^{(|V|^2)}) \cdot (p^{(|V|^2)})$.*

Proof. The difference of two simple lassos l_1 and l_2 of length at most $|V|$ can be represented by another simple lasso $l = l_1 \times l_2$ of length at most $|V|^2$. If the maximum length of the lassos is $|V|$, then the maximum length of the difference lasso will be $|V|^2$. Then, from Lemma 1 we immediately obtain that the maximum value of the denominator of the minimum non-zero difference of simple lassos is $(p^{(|V|^2)} - q^{(|V|^2)}) \cdot (p^{(|V|^2)})$. □

For notational convenience, let $\mathsf{bound}_W = (p^{|V|} - q^{|V|}) \cdot (p^{|V|})$ and $\mathsf{bound}_{\mathsf{diff}} = (p^{(|V|^2)} - q^{(|V|^2)}) \cdot (p^{(|V|^2)})$. Wlog, $|V| > 1$. Since, $\frac{1}{\mathsf{bound}_{\mathsf{diff}}} < \frac{1}{\mathsf{bound}_W}$, there is at most one rational number with denominator $\mathsf{bound}_{\mathsf{W}}$ or less in any interval of size $\frac{1}{\mathsf{bound}_{\mathsf{diff}}}$. Thus, if we can identify an interval of size less than $\frac{1}{\mathsf{bound}_{\mathsf{diff}}}$ around the optimal cost, then due to Corollary 1, the optimal cost will be the unique rational number with denominator $\mathsf{bound}_{\mathsf{W}}$ or less in this interval.

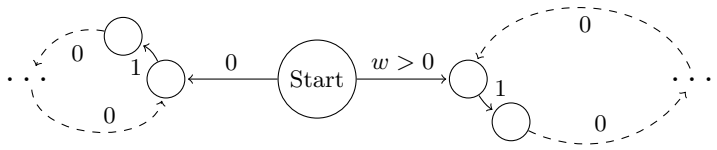

Fig. 1: Sketch of game graph which requires $\Omega(|V|^2)$ iterations

Thus, the final question is to identify a small enough interval (of size $\frac{1}{\mathsf{bound}_{\mathsf{diff}}}$ or less) such that the optimal cost lies within it. To find an interval around the optimal cost, we use a finite-horizon approximation of the optimal cost:

Lemma 2. *Let W be the optimal cost in quantitative game G. Let $\mu > 0$ be the maximum of absolute value of cost on transitions in G. Then, for all $k \in \mathbb{N}$,*

$$wt_k(v_{\mathsf{init}}) - \frac{1}{d^{k-1}} \cdot \frac{\mu}{d-1} \leq W \leq wt_k(v_{\mathsf{init}}) + \frac{1}{d^{k-1}} \cdot \frac{\mu}{d-1}$$

Proof. Since W is the limit of $wt_k(v_{\mathsf{init}})$ as $k \to \infty$, W must lie in between the minimum and maximum cost possible if the k-length game is extended to an infinite-length game. The minimum possible extension would be when the k-length game is extended by iterations in which the cost incurred in each round is $-\mu$. Therefore, the minimum possible value is $wt_k(v_{\mathsf{init}}) - \frac{1}{d^{k-1}} \cdot \frac{\mu}{d-1}$. Similarly, the maximum possible value is $wt_k(v_{\mathsf{init}}) + \frac{1}{d^{k-1}} \cdot \frac{\mu}{d-1}$. □

Now that we have an interval around the optimal cost, we can compute the number of iterations of VI required to make it smaller than $1/\mathsf{bound}_{\mathsf{diff}}$.

Theorem 1. *Let $G = (V, v_{\mathsf{init}}, E, \gamma)$ be a quantitative graph game. Let $\mu > 0$ be the maximum of absolute value of costs along transitions. The number of iterations required by the value-iteration algorithm is*

1. *$\mathcal{O}(|V|^2)$ when discount factor $d \geq 2$,*
2. *$\mathcal{O}\left(\frac{\log(\mu)}{d-1} + |V|^2\right)$ when discount factor $1 < d < 2$.*

Proof. As discussed earlier, the unique rational number with denominator $\frac{1}{\mathsf{bound}_W}$ or less within the interval $(wt_k(v_{\mathsf{init}}) - \frac{1}{d^{k-1}} \cdot \frac{\mu}{d-1}, wt_k(v_{\mathsf{init}}) + \frac{1}{d^{k-1}} \cdot \frac{\mu}{d-1})$ for a large enough $k > 0$ such that the interval's size is less than $\frac{1}{\mathsf{bound}_{\mathsf{diff}}}$ must be the optimal cost. Thus, our task is to determine the value of $k > 0$ such that $2 \cdot \frac{\mu}{d-1 \cdot d^{k-1}} \leq \frac{1}{\mathsf{bound}_{\mathsf{diff}}}$ holds. $d \geq 2$ is easy to simplify. $1 < d < 2$ involves approximations of logarithms of small values. Details in supplemental material. □

Lower bound on number of iterations. We establish a matching lower bound of $\Omega(|V|^2)$ iterations to show that our analysis is tight.

Consider the sketch of a quantitative game in Fig 1. Let all states belong to the maximizing player. Hence, the optimization problem reduces to searching for a *path* with optimal cost. Now let the loop on the right-hand side (RHS) be larger than the loop on the left-hand side (LHS). For carefully chosen values of

w and lengths of the loops, one can show that the path for optimal cost of a k-length game is along the RHS loop when k is small, but along the LHS loop when k is large. This way, the correct maximal value can be obtained only at a large value for k. Hence the VI algorithm runs for at least enough iterations that the optimal path will be in the LHS loop. By meticulous reverse engineering of the size of both loops and the value of w, one can guarantee that $k = \Omega(|V|^2)$. A concrete instance is presented in the supplemental material.

3.3 Worst-case complexity analysis of VI for optimization

Finally, we complete the worst-case complexity analysis of VI for optimization. We account for the the cost of arithmetic operations since they appear in abundance in VI. We demonstrate that there are orders-of-magnitude of difference in complexity under different models of arithmetic, namely unit-cost and bit-cost.

Unit-cost model. Under the unit-cost model of arithmetic, all arithmetic operations are assumed to take constant time.

Theorem 2. *Let $G = (V, v_{\mathsf{init}}, E, \gamma)$ be a quantitative graph game. Let $\mu > 0$ be the maximum of absolute value of costs along transitions. The worst-case complexity of the optimization problem under unit-cost model of arithmetic is*

1. $\mathcal{O}(|V|^2 \cdot |E|)$ *when discount factor* $d \geq 2$,
2. $\mathcal{O}\Big(\frac{\log(\mu)\cdot|E|}{d-1} + |V|^2 \cdot |E|\Big)$ *when discount factor* $1 < d < 2$.

Proof. The cost of the j-th iteration is $\mathcal{O}(E)$ since every transition is visited once. Thus, the complexity is $\mathcal{O}(|E|)$ multiplied by the number of iterations. □

Bit-cost model. Under the bit-cost model, the cost of arithmetic operations depends on the size of the numerical values. Integers are represented in their bit-wise representation. Rational numbers $\frac{r}{s}$ are represented as a tuple of the bit-wise representation of integers r and s. For two integers of length n and m, the cost of their addition and multiplication is $O(m + n)$ and $O(m \cdot n)$, respectively.

Theorem 3. *Let $G = (V, v_{\mathsf{init}}, E, \gamma)$ be a quantitative graph game. Let $\mu > 0$ be the maximum of absolute value of costs along transitions. Let $d = \frac{p}{q} > 1$ be the discount factor. The worst-case complexity of the optimization problem under the bit-cost model of arithmetic is*

1. $\mathcal{O}(|V|^4 \cdot |E| \cdot \log p \cdot \max\{\log \mu, \log p\})$ *when* $d \geq 2$,
2. $\mathcal{O}(\Big(\frac{\log(\mu)}{d-1} + |V|^2\Big)^2 \cdot |E| \cdot \log p \cdot \max\{\log \mu, \log p\})$ *when* $1 < d < 2$.

Proof (Sketch). Since arithmetic operations incur a cost and the length of representation of intermediate costs increases linearly in each iteration, we show (details in supplemental material) that the cost of conducting the j-th iteration is $\mathcal{O}(|E| \cdot j \cdot \log \mu \cdot \log p)$. Their summation will return the given expressions. □

Remarks on integer discount factor. Our analysis shows that when the discount factor is an integer ($d \geq 2$), VI requires $\Theta(|V|^2)$ iterations. Its worst-case complexity is, therefore, $\mathcal{O}(|V|^2 \cdot |E|)$ and $\mathcal{O}(|V|^4 \cdot |E|)$ under the unit-cost and bit-cost models for arithmetic, respectively. From a practical point of view, the bit-cost model is more relevant since implementations of VI will use multi-precision libraries to avoid floating-point errors. While one may argue that the upper bounds in Theorem 3 could be tightened, they would not improve significantly due to the $\Omega(|V|^2)$ lower bound on number of iterations.

3.4 Satisficing via value-iteration

We present our first algorithm for the satisficing problem. It is an adaptation of VI. However, we see that it does not fare better than VI for optimization.

VI-based algorithm for satisficing is described as follows: Perform VI for optimization. Terminate as soon as one of these occurs: (a). VI completes as many iterations from Theorem 1, or (b). The threshold value falls outside the interval defined in Lemma 2. Either way, one can tell how the threshold value relates to the optimal cost to solve satisficing. Clearly, (a) needs as many iterations as optimization; (b) does not reduce the number of iterations since it is inversely proportional to the distance between optimal cost and threshold value:

Theorem 4. *Let $G = (V, v_{\mathsf{init}}, E, \gamma)$ be a quantitative graph game with optimal cost W. Let $v \in \mathbb{Q}$ be the threshold value. Then number of iterations taken by a VI-based algorithm for the satisficing problem is* $\min\{O(|V|^2), \log \frac{\mu}{|W|-v}\}$ *if* $d \geq 2$ *and* $\min\{\mathcal{O}\Big(\frac{\log(\mu)}{d-1} + |V|^2\Big), \log \frac{\mu}{|W|-v}\}$ *if* $1 < d < 2$.

Observe that this bound is tight since the lower bounds from optimization apply here as well. The worst-case complexity can be completed using similar computations from § 3.3. Since, the number of iterations is identical to Theorem 1, the worst-case complexity will be identical to Theorem 2 and Theorem 3, showing no theoretical improvement. However, its implementations may terminate soon for threshold values far from the optimal but it will retain worst-case behavior for ones closer to the optimal. The catch is since the optimal cost is unknown apriori, this leads to a highly variable and *non-robust* performance.

4 Satisficing via Comparators

Our second algorithm for satisficing is based on purely automata-methods. While this approach operates with integer discount factors only, it runs linearly in the size of the quantitative game. This is lower than the number of iterations required by VI, let alone the worst-case complexities of VI. This approach reduces satisficing to solving a safety or reachability game using comparator automata.

The intuition is as follows: Given threshold value $v \in \mathbb{Q}$ and relation R, let the satisficing problem be to ensure cost of plays relates to v by R. Then, a play ρ is *winning for satisficing with v and* R if its cost sequence A satisfies $DS(A, d)$ R

v, where $d > 1$ is the discount factor. When d is an integer and $v = 0$, this simply checks if A is in the safety/co-safety comparator, hence yielding the reduction.

The caveat is the above applies to $v = 0$ only. To overcome this, we extend the theory of comparators to permit arbitrary threshold values $v \in \mathbb{Q}$. We find that results from $v = 0$ transcend to $v \in \mathbb{Q}$, and offer compact comparator constructions (§ 4.1). These new comparators are then used to reduce satisficing to develop an efficient and scalable algorithm (§ 4.2). Finally, to procure a well-rounded view of its performance, we conduct an empirical evaluation where we see this comparator-based approach outperform the VI approaches § 4.3.

4.1 Foundations of comparator automata with threshold $v \in \mathbb{Q}$

This section extends the existing literature on comparators with threshold value $v = 0$ [6,5,9] to permit non-zero thresholds. The properties we investigate are of safety/co-safety and ω-regularity. We begin with formal definitions:

Definition 3 (Comparison language with threshold $v \in \mathbb{Q}$). *For an integer upper bound $\mu > 0$, discount factor $d > 1$, equality or inequality relation $\mathsf{R} \in \{<, >, \leq, \geq, =, \neq\}$, and a threshold value $v \in \mathbb{Q}$ the* comparison language with upper bound μ, relation R, discount factor d and threshold value v *is a language of infinite words over the alphabet $\Sigma = \{-\mu, \ldots, \mu\}$ that accepts $A \in \Sigma^\omega$ iff $DS(A, d)$ R v holds.*

Definition 4 (Comparator automata with threshold $v \in \mathbb{Q}$). *For an integer upper bound $\mu > 0$, discount factor $d > 1$, equality or inequality relation $\mathsf{R} \in \{<, >, \leq, \geq, =, \neq\}$, and a threshold value $v \in \mathbb{Q}$ the* comparator automata with upper bound μ, relation R, discount factor d and threshold value v *is an automaton that accepts the DS comparison language with upper bound μ, relation R, discount factor d and threshold value v.*

Safety and co-safety of comparison languages. The primary observation is that to determine if $DS(A, d)$ R v holds, it should be sufficient to examine finite-length prefixes of A since weights later on get heavily discounted. Thus,

Theorem 5. *Let $\mu > 1$ be the integer upper bound. For arbitrary discount factor $d > 1$ and threshold value $v \in \mathbb{Q}$*

1. *Comparison languages are safety languages for relations $R \in \{\leq, \geq, =\}$.*
2. *Comparison language are co-safety languages for relations $R \in \{<, >, \neq\}$.*

Proof. The proof is identical to that for threshold value $v = 0$ from [9]. □

Regularity of comparison languages. We investigate when comparison languages with arbitrary threshold values are ω-regular. Prior work on threshold value $v = 0$ shows that a comparator is ω-regular iff the discount factor is an integer [7]. We show the same result holds with arbitrary threshold values $v \in \mathbb{Q}$.

First of all, trivially, comparators with arbitrary threshold value are not ω-regular for non-integer discount factors, since that already holds when $v = 0$.

The rest of this section proves ω-regularity with arbitrary threshold values for integer discount factors. But first, let us introduce some notations: Since $v \in \mathbb{Q}$, w.l.o.g. we assume that the it has an n-length representation $v = v[0]v[1]\dots v[m](v[m+1]v[m+2]\dots v[n])^\omega$. By abuse of notation, we denote both the expression $v[0]v[1]\dots v[m](v[m+1]v[m+2]\dots v[n])^\omega$ and the value $DS(v[0]v[1]\dots v[m](v[m+1]v[m+2]\dots v[n])^\omega, d)$ by v.

We will construct a Büchi automaton for the comparison language $\mathcal{L}_{\leq}$ for relation $\leq$, threshold value $v \in \mathbb{Q}$ and an integer discount factor. This is sufficient to prove ω-regularity for all relations since Büchi automata are closed.

From safety/co-safety of comparison languages, we argue it is sufficient to examine the discounted-sum of finite-length weight sequences to know if their infinite extensions will be in $\mathcal{L}_{\leq}$. For instance, if the discounted-sum of a finite-length weight-sequence W is *very large*, W could be a bad-prefix of $\mathcal{L}_{\leq}$. Similarly, if the discounted-sum of a finite-length weight-sequence W is *very small* then for all of its infinite-length bounded extensions Y, $DS(W \cdot Y, d) \leq v$. Thus, a mathematical characterization of *very large* and *very small* would formalize a criterion for membership of sequences in $\mathcal{L}_{\leq}$ based on their finite-prefixes.

To this end, we use the concept of a *recoverable gap* (or gap value), which is a measure of distance of the discounted-sum of a finite-sequence from 0 [12]. The recoverable gap of a finite weight-sequences W with discount factor d, denoted $\mathsf{gap}(W, d)$, is defined as follows: If $W = \varepsilon$ (the empty sequence), $\mathsf{gap}(\varepsilon, d) = 0$, and $\mathsf{gap}(W, d) = d^{|W|-1} \cdot DS(W, d)$ otherwise. Then, Lemma 3 formalizes *very large* and *very small* in Item 1 and Item 2, respectively, w.r.t. recoverable gaps. As for notation, given a sequence A, let $A[\dots i]$ denote its i-length prefix:

Lemma 3. *Let $\mu > 0$ be the integer upper bound, $d > 1$ be the discount factor. Let $v \in \mathbb{Q}$ be the threshold value such that $v = v[0]v[1]\dots v[m](v[m+1]v[m+2]\dots v[n])^\omega$. Let W be a non-empty, bounded, finite-length weight-sequence.*

1. $\mathsf{gap}(W - v[\cdots |W|], d) > \frac{1}{d} \cdot DS(v[|W| \cdots], d) + \frac{\mu}{d-1}$. *iff for all infinite-length, bounded extensions Y, $DS(W \cdot Y, d) > v$*
2. $\mathsf{gap}(W - v[\cdots |W|], d) \leq \frac{1}{d} \cdot DS(v[|W| \cdots], d) - \frac{\mu}{d-1}$ *iff For all infinite-length, bounded extensions Y, $DS(W \cdot Y, d) \leq v$*

Proof. We present the proof of one direction of Item 1. The others follow similarly. Let W be s.t. for every infinite-length, bounded extension Y, $DS(W \cdot Y, d) > v$ holds. Then $DS(W, d) + \frac{1}{d^{|W|}} \cdot DS(Y, d) \geq DS(v[\cdots |W|] \cdot v[|W| \cdots], d)$ implies $DS(W, d) - DS(v[\cdots |W|], d) > \frac{1}{d^{|W|}} \cdot (DS(v[|W| \cdots], d) - DS(Y, d))$ implies $\mathsf{gap}(W - v[\cdots |W|], d) > \frac{1}{d}(DS(v[|W| \cdots], d) + \frac{\mu \cdot d}{d-1})$. □

This segues into the state-space of the Büchi automaton. We define the state space so that state s represents the gap value s. The idea is that all finite-length weight sequences with gap value s will terminate in state s. To assign transition between these states, we observe that gap value is defined inductively as follows:

$\mathsf{gap}(\varepsilon, d) = 0$ and $\mathsf{gap}(W \cdot w, d) = d \cdot \mathsf{gap}(W, d) + w$, where $w \in \{-\mu, \ldots, \mu\}$. Thus there is a transition from state s to state t on $a \in \{-\mu, \ldots, \mu\}$ if $t = d \cdot s + a$.

Since $\mathsf{gap}(\varepsilon, d) = 0$, state 0 is assigned to be the initial state.

The issue with this construction is it has infinite states. To limit that, we use Lemma 3. Since Item 1 is a necessary and sufficient criteria for bad prefixes of safety language $\mathcal{L}_{\leq}$, all states with value larger than Item 1 are fused into one non-accepting sink. For the same reason, all states with gap value less than Item 1 are accepting states. Due to Item 2, all states with value less than Item 2 are fused into one accepting sink. Finally, since d is an integer, gap values are integral. Thus, there are only finitely many states between Item 2 and Item 1.

Formal construction. The formal construction of the Büchi automaton $\mathcal{A} = (S, s_I, \Sigma, \delta, \mathcal{F})$ for the comparison language for integer upper bound $\mu > 0$, integer discount factor $d > 1$, inequality relation $\leq$, and threshold value $v \in \mathbb{Q}$ s.t. $v = v[0]v[1] \ldots v[m](v[m+1]v[m+2] \ldots v[n])^\omega$ is given by where:

For $i \in \{0, \ldots, n\}$, let $\mathsf{U}_i = \frac{1}{d} \cdot DS(v[i \cdots], d) + \frac{\mu}{d-1}$ (Lemma 3, Item 1)

For $i \in \{0, \ldots, n\}$, let $\mathsf{L}_i = \frac{1}{d} \cdot DS(v[i \cdots], d) - \frac{\mu}{d-1}$ (Lemma 3, Item 2)

- States $S = \bigcup_{i=0}^{n} S_i \cup \{\mathsf{bad}, \mathsf{veryGood}\}$ where $S_i = \{(s, i) | s \in \{\lfloor \mathsf{L}_i \rfloor + 1, \ldots, \lfloor \mathsf{U}_i \rfloor\}\}$
- Initial state $s_I = (0, 0)$, Accepting states $\mathcal{F} = S \setminus \{\mathsf{bad}\}$
- Alphabet $\Sigma = \{-\mu, -\mu + 1, \ldots, \mu - 1, \mu\}$
- Transition function $\delta \subseteq S \times \Sigma \to S$ where $(s, a, t) \in \delta$ then:
  1. If $s \in \{\mathsf{bad}, \mathsf{veryGood}\}$, then $t = s$ for all $a \in \Sigma$
  2. If s is of the form (p, i), and $a \in \Sigma$
     (a) If $d \cdot p + a - v[i] > \lfloor \mathsf{U}_i \rfloor$, then $t = \mathsf{bad}$
     (b) If $d \cdot p + a - v[i] \leq \lfloor \mathsf{L}_i \rfloor$, then $t = \mathsf{veryGood}$
     (c) If $\lfloor \mathsf{L}_i \rfloor < d \cdot p + a - v[i] \leq \lfloor \mathsf{U}_i \rfloor$,
        i. If $i == n$, then $t = (d \cdot p + a - v[i], m + 1)$
        ii. Else, $t = (d \cdot p + a - v[i], i + 1)$

We skip proof of correctness as it follows from the above discussion. Observe, $\mathcal{A}$ is deterministic. It is a safety automaton as all non-accepting states are sinks.

Theorem 6. *Let $\mu > 0$ be an integer upper bound, $d > 1$ an integer discount factor, R an equality or inequality relation, and $v \in \mathbb{Q}$ the threshold value with an n-length representation given by $v = v[0]v[1] \ldots v[m](v[m+1]v[m+2] \ldots v[n])^\omega$.*

1. *The DS comparator automata for μ, d, R, v is ω-regular iff d is an integer.*
2. *For integer discount factors, the DS comparator is a safety or co-safety automaton with $\mathcal{O}(\frac{\mu \cdot n}{d-1})$ states.*

Proof. Item 1 is immediate from the above discussion. Item 2 is clear from the construction for inequality $\leq$. Since the comparator for $\leq$ it is a deterministic (safety automaton), the comparator for $>$ is obtained by simply flipping the accepting and non-accepting states. This is a co-safety automaton of the same size. One can argue similarly for the remaining relations. □

4.2 Satisficing via safety and reachability games

This section describes our comparator-based linear-time algorithm for satisficing for integer discount factors.

As described earlier, given discount factor $d > 1$, a play is winning for satisficing with threshold value $v \in \mathbb{Q}$ and relation R if its cost sequence A satisfies $DS(A, d)$ R v. We now know from Theorem 6, that the winning condition for plays can be expressed as a safety or co-safety automaton for any $v \in \mathbb{Q}$ as long as the discount factor is an integer. Therefore, a *synchronized product* of the quantitative game with the safety or co-safety comparator denoting the winning condition completes the reduction to a safety or reachability game, respectively.

Formal reduction. Let $G = (V = V_0 \uplus V_1, v_{\mathsf{init}}, E, \gamma)$ be a quantitative game, $d > 1$ the integer discount factor, R the equality or inequality relation, and $v \in \mathbb{Q}$ the threshold value with an n-length representation. Let $\mu > 0$ be the maximum of absolute values of costs along transitions in G. Then, the first step is to construct the safety/co-safety comparator $\mathcal{A} = (S, s_I, \Sigma, \delta, \mathcal{F})$ for μ, d, R and v. The next is to synchronize the product of G and $\mathcal{A}$ over weights to construct the game $\mathsf{GA} = (W = W_0 \cup W_1, s_0 \times \mathsf{init}, \delta_W, \mathcal{F}_W)$, where

- $W = V \times S$. In particular, $W_0 = V_0 \times S$ and $W_1 = V_1 \times S$. Since V_0 and V_1 are disjoint, W_0 and W_1 are disjoint too.
- Let $s_0 \times \mathsf{init}$ be the initial state of GA.
- Transition relation $\delta_W = W \times W$ is defined such that transition $((v, s), (v', s')) \in \delta_W$ synchronizes between transitions $(v, v') \in \delta$ and $(s, a, s') \in \delta_C$ if $a = \gamma((v, v'))$ is the cost of transition in G.
- $\mathcal{F}_W = V \times \mathcal{F}$. The game is a safety game if the comparator is a safety automaton and a reachability game if the comparator is a co-safety automaton.

Theorem 7. *Let $G = (V, v_{\mathsf{init}}, E, \gamma)$ be a quantitative game, $d > 1$ the integer discount factor, R the equality or inequality relation, and $v \in \mathbb{Q}$ the threshold value with an n-length representation. Let $\mu > 0$ be the maximum of absolute values of costs along transitions in G. Then,*

1. *The satisficing problem reduces to solving a safety game if $\mathsf{R} \in \{\leq, \geq\}$*
2. *The satisficing problem reduces to solving a reachability game if $\mathsf{R} \in \{<, >\}$*
3. *The satisficing problem is solved in $\mathcal{O}((|V| + |E|) \cdot \mu \cdot n)$ time.*

Proof. The first two points use a standard synchronized product argument.

We need the size of GA to analyze the worst-case complexity. Clearly, GA consists of $\mathcal{O}(|V| \cdot \mu \cdot n)$ states. To establish the number of transitions in GA, observe that every state (v, s) in GA has the same number of outgoing edges as state v in G because the comparator $\mathcal{A}$ is deterministic. Since GA has $\mathcal{O}(\mu \cdot n)$ copies of every state $v \in G$, there are a total of $\mathcal{O}(|E| \cdot \mu \cdot n)$ transitions in GA. Since GA is either a safety or a reachability game, it is solved in linear-time to its size. Thus, the overall complexity is $\mathcal{O}((|V| + |E|) \cdot \mu \cdot n)$. □

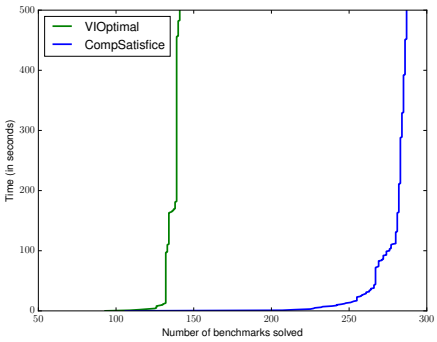

Fig. 2: Cactus plot. $\mu = 5, v = 3$. Total benchmarks = 291

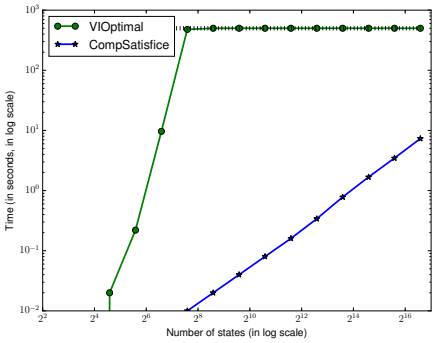

Fig. 3: Single counter scalable benchmark. $\mu = 5, v = 3$. Timeout = 500s.

With respect to the value μ, the VI-based solutions are logarithmic in the worst case, while comparator-based solution is linear due to the size of the comparator. From a practical perspective, this may not be a limitation since weights along transitions can be scaled down. The parameter that cannot be altered is the size of the quantitative game. With respect to that, the comparator-based solution displays clear superiority.

4.3 Implementation and Empirical Evaluation

The goal of the empirical analysis is to determine whether the practical performance of these algorithms resonate with our theoretical discoveries.

For an apples-to-apples comparison, we implement three algorithms: (a) VIOptimal: Optimization via value-iteration, (b)VISatisfice: Satisficing via value-iteration, and (c). CompSatisfice: Satisficing via comparators. All tools have been implemented in C++. To avoid floating-point errors in VIOptimal and VISatisfice, the tools invoke the open-source GMP (GNU Multi-Precision) [2]. Since all arithmetic operations in CompSatisfice are integral only, it does not use GMP.

To avoid completely randomized benchmarks, we create ∼290 benchmarks from $\mathsf{LTL_f}$ benchmark suite [27]. The state-of-the-art $\mathsf{LTL_f}$-to-automaton tool Lisa [8] is used to convert $\mathsf{LTL_f}$ to (non-quantitative) graph games. Weights are randomly assigned to transitions. The number of states in our benchmarks range from 3 to 50000+. Discount factor $d = 2$, threshold $v \in [0 - 10]$. Experiments were run on 8 CPU cores at 2.4GHz, 16GB RAM on a 64-bit Linux machine.

Observations and Inferences[1] Overall, we see that VISatisfice is efficient and scalable, and exhibits steady and predictable performance.

CompSatisfice *outperforms* VIOptimal in both runtime and number of benchmarks solved, as shown in Fig 2. It is crucial to note that all benchmarks solved

[1] Figures are best viewed online and in color

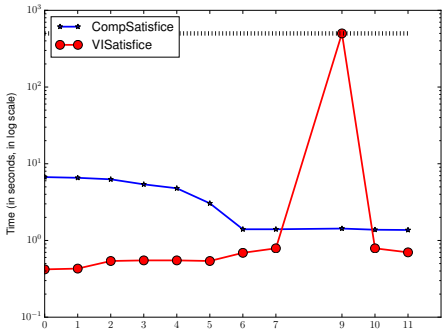

Fig. 4: Robustness. Fix benchmark, vary v. $\mu = 5$. Timeout = 500s.

by VIOptimal had fewer than 200 states. In contrast, CompSatisfice solves much larger benchmarks with 3-50000+ number of states.

To test scalability, we compared both tools on a set of scalable benchmarks. For integer parameter $i > 0$, the i-th scalable benchmark has $3 \cdot 2^i$ states. Fig 3 plots number-of-states to runtime in log-log scale. Therefore, the slope of the straight line will indicate the degree of polynomial (in practice). It shows us that CompSatisfice exhibits linear behavior (slope $\sim$1), whereas VIOptimal is much more expensive (slope $>>$ 1) even in practice.

CompSatisfice *is more robust than* VISatisfice. We compare CompSatisfice and VISatisfice as the threshold value changes. This experiment is chosen due to Theorem 4 which proves that VISatisfice is non-robust. As shown in Fig 4, the variance in performance of VISatisfice is very high. This is because its worst-case complexity is unavoidable in practice. On that other hand, CompSatisfice stays steady in performance owning to its low complexity.

5 Temporally extended goals and broader applicability

Having witnessed algorithmic improvements of comparator-based satisficing over VI-based algorithms, we now shift focus to the question of applicability. While this section examines this with respect to the ability to extend to temporal goals, this discussion highlights a core strength of comparator-based reasoning in satisficing and shows its promise in a broader variety of problems.

The problem of extending optimal/satisficing solutions with a temporal goal is to determine whether there exists an optimal/satisficing solution that also satisfies a given temporal goal. Formally, given a quantitative game G, a labelling function $\mathcal{L} : V \to 2^{AP}$ which assigns states V of G to atomic propositions from the set AP, and a temporal goal φ over AP, we say a *play* $\rho = v_0 v_1 \dots$ *satisfies* φ if its proposition sequence given by $\mathcal{L}(v_0)\mathcal{L}(v_1)\dots$ satisfies the formula φ. Then to solve *optimization/satisficing with a temporal goal* is to determine if there exists a solutions that is optimal/satisficing and also satisfies the temporal goal along resulting plays. Prior work has proven that the optimization problem

cannot be extended to temporal goals [13] unless the temporal goals are very simple safety properties [10,29]. In contrast, our comparator-based solution for satisficing can naturally be extended to temporal goals, in fact to all ω-regular properties, owing to its automata-based underpinnings, as shown below:

Theorem 8. *Let G a quantitative game with state set V, $\mathcal{L} : V \to 2^{AP}$ be a labelling function over set of atomic propositions AP, and φ be a temporal goal over AP and $\mathcal{A}_\varphi$ be its equivalent deterministic parity automaton. Let $d > 1$ be an integer discount factor, μ be the maximum of the absolute values of costs along transitions, and $v \in Q$ be the threshold value with an n-length representation. Then, solving satisficing with temporal goals reduces to solving a parity game of size linear in $|V|$, μ, n and $|\mathcal{A}_\varphi|$.*

Proof. The reduction involves two steps of synchronized products. The first reduces the satisficing problem to a safety/reachability game while preserving the labelling function. The second synchronization product is between the safety/reachability game with the DPA $\mathcal{A}_\varphi$. These will synchronize on the atomic propositions in the labeling function and DPA transitions, respectively. Therefore, resulting parity game will be linear in $|V|$, μ and n, and $|\mathcal{A}_\varphi|$. □

Broadly speaking, our ability to solve satisficing via automata-based methods is a key feature as it propels a seamless integration of quantitative properties (threshold bounds) with qualitative properties, as both are grounded in automata-based methods. VI-based solutions are inhibited to do so since numerical methods are known to not combine well with automata-based methods which is so prominent with qualitative reasoning [5,19]. This key feature could be exploited in several other problems to show further benefits of comparator-based satisficing over optimization and VI-based methods.

6 Concluding remarks

This work introduces the satisficing problem for quantitative games with discounted sum cost-model. When the discount factor is an integer, we present a comparator-based solution for satisficing that exhibits algorithmic improvements - better complexity and efficient, scalable, and robust performance - and broader applicability over traditional solutions based on numerical for satisficing and optimization. Other technical contributions include presentation of the missing proof of value-iteration for optimization and extension of the literature on comparator automata to arbitrary threshold values.

An undercurrent of our comparator-based approach for satisficing is that it offers an automata-based replacement to traditional numerical methods. By doing so, it paves a way to combine quantitative and qualitative reasoning without compromising on theoretical guarantees and even performance. The affirmative outcomes of this work encourage us to tackle more challenging problems in this space, possibly with more complex environments, variability in information, and their combinations.

References

1. Satisficing. `https://en.wikipedia.org/wiki/Satisficing`.
2. GMP. `https://gmplib.org/`.
3. B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed computing*, 2(3):117–126, 1987.
4. C. Baier. Probabilistic model checking. In *Dependable Software Systems Engineering*, pages 1–23. 2016.
5. S. Bansal, S. Chaudhuri, and M. Y. Vardi. Automata vs linear-programming discounted-sum inclusion. In *Proc. of International Conference on Computer-Aided Verification (CAV)*, 2018.
6. S. Bansal, S. Chaudhuri, and M. Y. Vardi. Comparator automata in quantitative verification. In *Proc. of International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, 2018.
7. S. Bansal, S. Chaudhuri, and M. Y. Vardi. Comparator automata in quantitative verification (full version). *CoRR*, abs/1812.06569, 2018.
8. S. Bansal, Y. Li, L. Tabajara, and M. Y. Vardi. Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications. In *Proc. of AAAI*, 2020.
9. S. Bansal and M. Y. Vardi. Safety and co-safety comparator automata for discounted-sum inclusion. In *Proc. of International Conference on Computer-Aided Verification (CAV)*, 2019.
10. J. Bernet, D. Janin, and I. Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications*, 36(3):261–275, 2002.
11. R. Bloem, K. Chatterjee, T. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *Proc. of CAV*, pages 140–156. Springer, 2009.
12. U. Boker and T. A. Henzinger. Exact and approximate determinization of discounted-sum automata. *LMCS*, 10(1), 2014.
13. K. Chatterjee, T. A. Henzinger, J. Otop, and Y. Velner. Quantitative fair simulation games. *Information and Computation*, 254:143–166, 2017.
14. D. Clark, S. Hunt, and P. Malacaria. A static analysis for quantifying information flow in a simple imperative language. *Journal of Computer Security*, 15(3):321–371, 2007.
15. B. Finkbeiner, C. Hahn, and H. Torfah. Model checking quantitative hyperproperties. In *Proc. of CAV*, pages 144–163. Springer, 2018.
16. T. D. Hansen, P. B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM*, 60, 2013.
17. K. He, M. Lahijanian, L. Kavraki, and M. Vardi. Reactive synthesis for finite tasks under resource constraints. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 5326–5332. IEEE, 2017.
18. O. Kupferman and M. Y. Vardi. Model checking of safety properties. In *Proc. of CAV*, pages 172–183. Springer, 1999.
19. M. Kwiatkowska. Quantitative verification: Models, techniques and tools. In *Proc. 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 449–458. ACM Press, September 2007.
20. M. Kwiatkowska, G. Norman, and D. Parker. Advances and challenges of probabilistic model checking. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1691–1698. IEEE, 2010.

21. M. Lahijanian, S. Almagor, D. Fried, L. Kavraki, and M. Vardi. This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction. In *AAAI*, pages 3664–3671, 2015.
22. M. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1994.
23. M. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.
24. S. A. Seshia, A. Desai, T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, S. Shivakumar, M. Vazquez-Chanlatte, and X. Yue. Formal specification for deep neural networks. In *Proc. of ATVA*, pages 20–34. Springer, 2018.
25. L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095, 1953.
26. R. Sutton and A. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.
27. L. M. Tabajara and M. Y. Vardi. Partitioning techniques in LTLf synthesis. In *IJCAI*, pages 5599–5606. AAAI Press, 2019.
28. W. Thomas, T. Wilke, et al. *Automata, logics, and infinite games: A guide to current research*, volume 2500. Springer Science & Business Media, 2002.
29. M. Wen, R. Ehlers, and U. Topcu. Correct-by-synthesis reinforcement learning with temporal logic constraints. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4983–4990. IEEE, 2015.
30. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1):343–359, 1996.