Restricted Flow Games

Ravid Alon[0009−0005−0150−9620] and and Orna Kupferman[0000−0003−4699−6117]

School of Computer Science and Engineering, The Hebrew University, Israel

Abstract. Classical *graph problems* are defined with respect to plain graphs, namely vertices connected by possibly weighted edges. On the other hand, *model checking* studies rich graph structure and semantics, in particular *labeled graphs* and *game graphs*, which model involved realistic settings. Extending classical graph problems to graphs with a rich semantics offers an interesting and fresh perspective for classical graph algorithms. In addition, it extends the applicability of graph algorithms to rich settings.

In the classical *maximum-flow* problem, the goal is to find the maximal amount of flow that can be transferred through a network, by directing the flow in each vertex into outgoing edges. The problem has been recently extended to labeled graphs and game graphs. We introduce and study *restricted flow games*, an extension of the maximum flow-problem to graphs that are both labeled and game graphs. In these games, the edges of the network are labeled by letters over some alphabet, and the vertices are partitioned between two players, the authority and the environment. Each player directs the flow entering her vertices to outgoing edges. The goal of the authority is to maximize the amount of flow that reaches the target along routes that satisfy a given specification – a language over the alphabet of labels. We study several aspects of restricted flow game as well as the complexity of decision problems on them.

1 Introduction

Graphs are used to model many types of relations and processes in physical, biological, social, and information systems. Many practical problems can be reduced to problems about graphs, making graph theory a preeminent research area in theoretical computer science [10, 12]. Different settings call for different types of graphs. For example, the edges of the graph may be *directed* or *weighted*, say for modeling lengths or costs. In many applications, the edges of the graph carry further information. For example, an edge may be associated with an action (e.g., in planning or in VLSI design), a query (e.g., in databases), properties like its provider or its security level (e.g., in a network of channels), and many more. For such applications, we need *labeled graphs*, in which each vertex or edge is labeled by a letter from some alphabet. Further, in some applications, vertices and edges may be controlled by different entities (e.g., in the modeling of reactive or multi-agent systems, where transitions depend on actions taken by the underlying entities), giving rise to *game graphs*.

Labeled graphs and game graphs have been extensively researched in the context of *model checking* [8]. A model-checking algorithm gets as input a model of the system, where vertices correspond to the configurations of the system and edges correspond to transitions between configurations. Given a specification for the system, the algorithm decides whether the model satisfies the specification. The graph algorithms that model checking uses are fairly basic, and evolve around reachability [33] or partitioning to strongly connected components [7, 32]. They are applied, however, to graphs with a rich structure and semantics. In comparison, graph algorithms solve rich problems on basic graphs. A recent research direction is a study of extensions of classical graph algorithms to labeled graphs and game graphs [23].

An example to an extension of a classical graph-theory problem to labeled graphs are *regular path queries*, where we find all pairs of vertices connected by a path such that the word read along the path satisfies a specification [28, 6]. Then, rather than finding any *shortest path* between two given vertices in a graph [10], it is sometimes desirable, say in transportation planning [5], web searching [1], or network routing, to restrict attention to paths that satisfy some constraint [4]. As a more elaborated example, the *constrained Eulerian path* problem is the problem of deciding whether a labeled graph has an Eulerian path that satisfies a given specification [25].

As for game graphs, for the basic problem of reachability, the two-player setting gives rise to *alternating graph reachability* [9]. As a more complex example, consider the general setting in which two players alternately claim edges of a graph G while making sure the graph they build together satisfies some monotone decreasing property. The *Turán numbers* and *Saturation numbers* refer to the number of edges that can be claimed while the property is maintained [15, 20]. Then, *spanning-tree games* are an extension of the spanning-tree problem [21]. In spanning-tree games, two players alternate turns constructing a spanning tree of a given connected weighted graph. In each turn, a player chooses an edge that does not close a circle. The game ends when the chosen edges form a spanning tree. The goal of the *min player* is to minimize the total weight of edges in the obtained spanning tree, while the goal of the *max player* is to maximize it.

The extensions of graph algorithms to rich settings typically have a computational cost. For example, reachability is in NLOGSPACE, while regular path queries are only known to be in P [28], and alternating reachability is P-complete; the classic Eulerian path problem can be solved in polynomial time, while the constrained variant is NP-complete [25]; finding a max spanning-tree can be done in polynomial time, whereas even evaluating a given strategy for the max player in a spanning-tree game is NP-hard [21].

A classical problem to which both the label-graph and game-graph extensions have been applied is the *maximum-flow problem* [10, 17]. A *flow network* is a directed graph where each edge has a capacity. The capacity is a bound on the amount of flow that can go through each edge. The amount of flow entering a vertex must be the same as the amount of flow leaving the vertex, except for the *source* vertex, which only has outgoing flow, and the *target* vertex, which

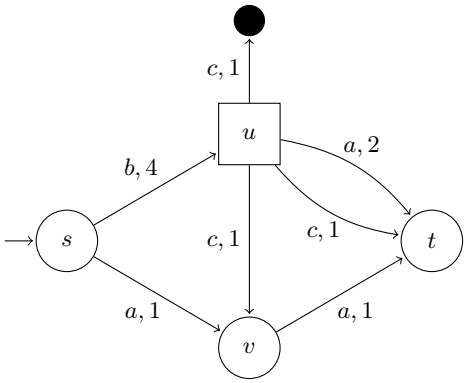
only has incoming flow. The maximum-flow problem is, given a flow network with source and target vertices, to determine the maximum flow that can go from the source to the target. The maximum-flow problem was first studied in the 1950s [13, 14], and has been researched extensively since then [2, 11, 18], with applications in routing, scheduling, and more.

Flow networks on labeled graphs were introduced by means of *capacitated automata* with a *utilization semantics* [24, 3]. Capacitated automata are finite word automata in which each transition has a capacity. In the utilization semantics of capacitated automata, the automata are viewed as accepting multiple words simultaneously, while respecting the capacity of each transition. Thus, an automaton mutually accepts a *multiset* of words if there is a multiset of accepting runs on these words, such that the number of times each transition is traversed by the runs is at most its capacity. The set of multisets mutually accepted by a capacitated automaton is its *multi-language*. The *maximum restricted-utilization* problem gets as input a capacitated automaton and a specification, and searches for the biggest multiset of words in the multi-language that contains only words that satisfy the specification. This problem is equivalent to an extension of the maximum-flow problem to labeled graphs. Indeed, capacitated automata are equivalent to labeled graphs, with states as vertices and capacitated transitions as labeled edges. In addition, each unit of flow in the network corresponds to a word, and so a flow in the network corresponds to a multiset of words. Thus, maximizing the number of words that are mutually accepted is the same as maximizing the flow that reaches the target along paths that satisfy the specification. It is shown in [24] that the maximum restricted-utilization problem is APX-complete. Thus, it is NP-complete, and there exists a constant c such that it is already NP-hard to approximate it within a multiplicative factor of c. When the specification contains only words of length at most 2, the problem can be solved in polynomial time.

Flow networks in a game setting were introduced and studied in [27, 19]. In *flow games*, the vertices of a flow network are partitioned between two players. Each player controls the flow through her vertices, by directing flow entering her vertices to outgoing edges. The goal of Player 0, the authority, is to maximize the total flow that reaches the target source, while Player 1, the environment, tries to minimize the flow. A *policy* for a vertex maps incoming flow to a function describing how the incoming flow is partitioned between the edges leaving that vertex. A policy for the source vertex assigns a flow to each outgoing edge, bounded by the edge's capacity. A strategy for a player consists of policies for all of her vertices. We consider *acyclic* flow networks. Then, given strategies for both players, calculating the flow in the game can be done in polynomial time. Finding the maximal flow in the game, as well as the optimal strategy for Player 0, is Σ_2^P-complete, and it is already Σ_2^P-hard to approximate. In the *unfortunate-flow problem* [26], the edges leaving the source are saturated, and all vertices besides the source are controlled by Player 1. The problem of finding the maximum flow in this case is co-NP-hard. Thus, the setting corresponds to evacuation scenarios, where we want to analyze the amount of flow that is guaranteed to reach the

target when the most unfortunate routing decisions are made. In the multi-player variant [19] of flow games, the vertices are partitioned between multiple players. Each player has a target vertex, and she tries to maximize the flow entering her own target. A *Nash Equilibrium* in a multi-player flow game is a set of strategies, one for each player, such that for every player, changing her own strategy does not increase the flow entering her target vertex. There exist multi-player flow games with no Nash Equilibrium, and the problem of deciding whether there exists a Nash Equilibrium in a given multi-player flow game is Σ_2^P-complete.

Example 1. Consider the labeled flow network N in the figure below. We represent vertices of Player 0 by circles and vertices of Player 1 by squares. Sinks are represented by filled circles. The labels are defined over the alphabet $\Sigma = \{a, b, c\}$. In the classical max-flow problem, the max flow in the network is 4. Indeed, 3 units of flow can travel via u, and 1 unit of flow can travel via v.



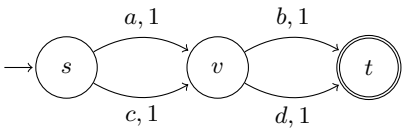
Consider the capacitated automaton obtained from N if we ignore the partition of the vertices between the players and the sink. Its language consists of the multisets $\{aa, ba, ba, bc\}$ and $\{bca, ba, ba, bc\}$. Consider a specification $(a + b)^*$. In the maximum restricted-utilization problem with this specification, the network needs to process only words that contain only a's and b's, and can thus process the multiset $\{aa, ba, ba\}$, inducing a flow of 3 units. For the specification $\Sigma^* \cdot c \cdot \Sigma^*$, where only words that contain c are needed, the network can process only 2 units of flow, induced by the multiset $\{bca, bc\}$.

Now, if we ignore the labels and analyze N as a flow game, a strategy for Player 0 saturates all edges leaving the source, so 5 units of flow leave s. An optimal strategy for Player 1 then direct one unit of flow entering u to a sink, one unit to v, and two units to t. The incoming flow to v is 2, while its outgoing capacity is 1. Thus, one unit of flow is lost, and one is directed to t. The flow in this case is 3, and it is the maximal flow Player 0 can ensure. □

We introduce and study *restricted flow games* (RFGs), an extension of flow games to graphs that are both labeled and game graph. As in flow games, the vertices in an RFG are partitioned between two players, and each player directs

the flow entering the vertices in her control. The input in RFGs includes a specification. The goal of Player 0, the authority, is to maximize the number of units of flow that reach the target vertex via paths that induce words that satisfy the specification. Player 1 corresponds to the environment, and tries to minimize the number of such units of flow, either by directing them into sinks (namely, vertices that have no outgoing edges), into vertices whose incoming flow is greater than their outgoing capacity, or through paths that do not satisfy the specification.

The strategies of the players in RFGs are the same as in flow games. Thus, they consist of policies for the vertices. Consequently, strategies are independent of the "history" of the units of flow entering their vertices, i.e., the path each unit of flow has traversed before reaching the vertex. While this enables a compact representation of strategies, it creates ambiguity in the path each unit of flow traverses. Indeed, given the incoming flow to some vertex, the player controlling this vertex can direct the flow into different outgoing edges, but we cannot determine which unit of flow goes to which outgoing edge. For example, consider the RFG below, and assume the edges leaving s are saturated.



Since the incoming flow to v is 2, the strategy for Player 0 saturates both outgoing edges. However, the strategy does not determine whether the flow traversing the a-labeled edge is directed to the b-labeled edge or the d-labeled edge, and so, its path might correspond to either ab or ad. Respectively, the path starting in the c-labeled edge might be labeled by either cd or cb.

In order to address this, we define two variants of RFGs: the maximal variant, which considers the "best-case scenario" for Player 0, and the minimal variant, which considers the "worst-case scenario" for Player 0. Formally, given a flow in the network $f : E \to \mathbb{N}$, a *decomposition* of the flow is a multiset of paths P from the source to the target in the network, such that the number of times each edge is traversed by paths in P is at most the flow through the edge. A decomposition P is *complete* if for every edge entering the target vertex, the number of times the edge is traversed in P is equal to the flow through the edge. Each decomposition of f is a possible outcome of the players directing the flow. Given a decomposition P, we consider the multiset of words $\ell(P)$ corresponding to the paths in P. Thus, in the example above, there are two complete decompositions P_1 and P_2, with $\ell(P_1) = \{ab, cd\}$ and $\ell(P_2) = \{ad, cb\}$. In the max variant, the outcome is the decomposition that maximizes the number of words in the specification. In the min variant, the outcome is the complete decomposition that minimizes this number.

Example 2. Consider an RFG played on the flow network shown in Example 1 with the specification $(a + b)^*$. If Player 0 and Player 1 play as described in

Example 1, then Player 1 needs to decide through which edge from u to t to direct the flow. An optimal strategy for Player 1 directs one unit of flow through each outgoing edge. Thus, one unit of flow traverses the c-labeled edge, and does not satisfy the specification. The other unit of flow entering u is directed to the a-labeled edge, and corresponds to the word ba, which satisfies the specification.

The incoming flow to v is 2, and so one unit of flow is lost and one is directed to t. Since Player 0 does not know the history of the flow entering v, we do not know which of the two units is lost and which is directed to t. In the maximal variant, the best case for Player 0 is considered, and so the unit of flow entering v from s is directed to t, and corresponds to the word aa, which satisfies the specification. Thus, the flow in the maximal variant is 2. Alternatively, in the minimal variant, the flow reaching v from s is lost, while the unit of flow reaching v from u is directed to t, but corresponds to a word not satisfying the specification. Thus, the flow in the minimal variant is 1. □

Analyzing the complexity of RFGs, we consider specifications given by automata and distinguish between different branching modes of the automaton: deterministic (DFA, which has a single run on every input word), nondeterministic (NFA, which may have several runs, and one of them has to be accepting), or universal (UFA, which may have several runs, all have to be accepting).

We start by studying the complexity of calculating the outcome of an RFG. That is, given strategies for both players, we calculate the number of words that satisfy the specification in the maximal and minimal decompositions. For this problem, as well as all subsequent problems we study, we consider the corresponding decision problem, parameterized by a threshold.

As mentioned above, calculating the flow in the game can be done in polynomial time. However, finding the maximal and minimal decompositions requires an additional computation. We show that for all types of automata, calculating the outcome is NP-complete for the maximal variant, and is co-NP-complete for the minimal variant. We proceed to consider the problem of determining the *positivity* of the outcome, namely whether the maximal or minimal outcome is positive or is 0. This is the special case of the general problem when the threshold is 1, and is of interest in settings where we do not care about loss of flow as long as at least one unit of flow is guaranteed to reach the target. We show that in the maximal variant, determining whether the outcome is positive is NP-complete when the specification is given by a UFA, but becomes NLOGSPACE-complete when it is given by a DFA or an NFA. In the minimal variant, determining the positivity is co-NP-complete for all types of automata.

We continue and study the complexity of the main problem in the context of RFG, namely finding the value of a given game in the maximal and minimal variants, as well as an optimal strategy for Player 0. That is, given an RFG, we find the strategy for Player 0 that ensures the best outcome against Player 1's best response, and the outcome in that case. The differences in the complexity of calculating the outcome and its positivity carry over to this problem: NP-completeness translates to an additional quantifier alternation and a "step up" in the polynomial hierarchy, compared to the Σ_2^P-completeness in flow games,

whereas co-NP-completeness does not. Thus, finding the flow is Σ_3^P-complete in the maximal variant, and is Σ_2^P-complete in the minimal variant. Similarly, when considering the problem of determining the positivity of the value, the maximal universal variant is Σ_3^P-complete, while all other variants are Σ_2^P-complete.

Our results, as well as directions for future research, are summarized in Section 5.

2 Preliminaries

2.1 Automata

An *automaton* is $\mathcal{A} = \langle \Sigma, Q, Q_0, \Delta, F \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $F \subseteq Q$ is a set of final states. If $|Q_0| = 1$, and for all $q \in Q$ and $\sigma \in \Sigma$, there is at most one $q' \in Q$ such that $\langle q, \sigma, q' \rangle \in \Delta$, then we say that $\mathcal{A}$ is deterministic, or a DFA, for short. The size of $\mathcal{A}$ is given by $|Q|$.

A *run* of $\mathcal{A}$ on a word $w = w_1 \cdots w_n \in \Sigma^*$ is $r = q_0, q_1, \ldots, q_n$, such that $q_0 \in Q_0$ and for all $0 \leq i < n$, we have that $\langle q_i, w_{i+1}, q_{i+1} \rangle \in \Delta$. We say that r is *accepting* if $q_n \in F$. Let $w \in \Sigma^*$. Note that if $\mathcal{A}$ is deterministic, then there is a unique run of $\mathcal{A}$ on w. In this case, we say that w is accepted by $\mathcal{A}$ if the unique run of $\mathcal{A}$ on w is accepting. Alternatively, $\mathcal{A}$ may be *nondeterministic* or *universal*, denoted NFA and UFA, respectively. If $\mathcal{A}$ is nondeterministic, we say that w is accepted by $\mathcal{A}$ if there *exists* an accepting run of $\mathcal{A}$ on w. If $\mathcal{A}$ is universal, we say that w is accepted by $\mathcal{A}$ if *every* run of $\mathcal{A}$ on w is accepting. The language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words in Σ^* that are accepted by $\mathcal{A}$.

2.2 Flow Games

A *flow network* is a tuple $N = \langle V, E, c, s, t \rangle$, where V is a set of vertices, $E \subseteq V \times V$ is a set of directed edges, $c : E \to \mathbb{N}$ is a capacity function, and $s, t \in V$ are source and target vertices. We assume that t is reachable from s and that the capacities are given in unary. A *flow game* [27] is $\mathcal{G} = \langle V_0, V_1, E, c, s, t \rangle$, such that $\langle V_0 \cup V_1, E, c, s, t \rangle$ is an acyclic flow network whose vertices are partitioned between two players, denoted Player 0 and Player 1. Each player controls the vertices in its vertex set. Player 0 tries to maximize the flow in the network and Player 1 tries to minimize the flow.

For a vertex $u \in V$, let E^u and E_u be the sets of incoming and outgoing edges to and from u, respectively. That is, $E^u = (V \times \{u\}) \cap E$ and $E_u = (\{u\} \times V) \cap E$. A *policy* for a vertex $u \in V$, for $u \neq s$, is a function that partitions an incoming flow between the outgoing edges. Formally, a policy for u is a function $f_u : \mathbb{N} \to \mathbb{N}^{E_u}$ such that for every flow $x \in \mathbb{N}$ and edge $e \in E_u$, we have $f_u(x)(e) \leq c(e)$ and $\sum_{e \in E_u} f_u(x)(e) = \min\{x, \sum_{e \in E_u} c(e)\}$. Thus, $f_u(x)$ assigns to each edge outgoing from u a flow that is bounded by its capacity. Also, when the incoming flow is larger than the capacity of the outgoing edges (which bounds the outgoing

flow), then flow is *lost* and the outgoing flow is lower than the incoming flow. In practice, loss of flow may correspond to leaks – fluid in a pipe system that is lost when the system is overflowed, to traffic that gets stuck – in jammed road systems, or to packets that are thrown by routers all whose outgoing channels are filled. Note that this is different from the traditional definition of flow in a network, which corresponds to the case all vertices belong to Player 0, and in which the "flow conservation" property is respected. Note also that $f_u(0)(e) = 0$ for every e. For the source vertex s, a policy is a function $f_s \in \mathbb{N}^{E_s}$ such that for every edge $e \in E_s$, we have $f_s(e) \leq c(e)$.

A *flow* in a flow game is a function $f \in \mathbb{N}^E$ that assigns to each edge the flow that travels in it. We require that for every edge $e \in E_u$, we have $f(e) \leq c(e)$, and for every vertex $u \in V$, except for s and t, we have $\sum_{e \in E_u} f(e) = \min\{\sum_{e \in E^u} f(e), \sum_{e \in E_u} c(e)\}$. That is, the flow in each edge is bounded by its capacity, and the flow that leaves each vertex is the minimum between the flow that enters the vertex and the sum of the capacities of edges outgoing from it. We focus on the case where the graph $\langle V, E \rangle$ is acyclic. Then, given policies f_u for all vertices in $u \in V$, we can calculate the flow in the game as follows. First, we order the vertices in a topological ordering. Then, we start from the vertex s (we ignore every vertex preceding s), and use f_s to assign a flow to each edge in E_s. Now, we continue to the next vertex in the topological ordering. Whenever we reach a vertex u, the incoming flow to u, denoted x, has already been calculated. We then use $f_u(x)$ to assign a flow for each edge in E_u, and continue along the topological ordering until we reach t. Since the flow that enters a vertex u depends only on the sub-game that reaches u, it is easy to see that the calculation above is independent of the topological ordering. Indeed, if u_1 and u_2 are not ordered, then flow that leaves u_1 does not reach u_2, and vice versa.

A strategy for Player i is a collection of policies, one for each vertex in V_i. Let F_0 and F_1 be the sets of all possible strategies of Player 0 and Player 1 respectively. Given strategies $\alpha \in F_0$ and $\beta \in F_1$, the flow in the game, denoted $f^{\alpha,\beta}$, can be calculated in polynomial time as described above.

2.3 Restricted Flow Games

A *restricted flow game*, RFG for short, is $\mathcal{G} = \langle \Sigma, V_0, V_1, E, c, s, t, L \rangle$, and it extends a flow game by a finite alphabet Σ and a regular language $L \subseteq \Sigma^*$, which we call the *specification*. In addition, the transition relation is $E \subseteq V \times \Sigma \times V$, thus each edge is labeled by a letter in Σ. Now, the goal of Player 0 is to maximize the number of units of flow that reach t, while traversing a path labeled by a word in L. That is, every unit of flow that reaches t is associated with the labels of the edges it traversed on its path from s to t. These labels are concatenated and form a word in Σ^*. Player 0 tries to maximize the number of such words that are in L.

We define the size of the graph of $\mathcal{G}$ as $\sum_{e \in E} c(e)$. Note that our definition corresponds to a representation of the graph with capacities given in unary. The specification L is given by an automaton $\mathcal{A}$ over the alphabet Σ, thus $L = L(\mathcal{A})$.

The size of the specification is defined to be the size of $\mathcal{A}$, and the size of $\mathcal{G}$ is the sum of the sizes of its graph and its specification. We indicate the type of $\mathcal{A}$ with the letter $\lambda \in \{D, N, U\}$, indicating whether it is a DFA, NFA, or UFA.

The definitions of policies, strategies, and their outcome are similar to their definitions in unrestricted flow games, except that now, $E^u = E \cap (V \times \Sigma \times \{u\})$ and $E_u = E \cap (\{u\} \times \Sigma \times V)$. Note that possibly $\langle v, \sigma_1, u\rangle, \langle v, \sigma_2, u\rangle \in E$ for $\sigma_1 \neq \sigma_2$, and the function c assigns capacity to labeled edges.

Note that policies determine the outgoing flow based on the incoming flow alone. The *history* of the flow, namely, the path traversed by each unit of flow, is unknown. A representation of policies that do depend on the history of the incoming flow is more complex (see discussion in Section 5.1). Our setting, where we give up dependency in the history result in simpler strategies but creates ambiguity in determining the outcome of the game, which we discuss below.

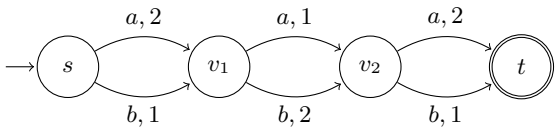
Given an edge $e = (v_1, \sigma, v_2) \in E$, we denote by $\ell(e) = \sigma$ the label of the edge. We expand ℓ to paths. Thus, given a path $\rho = e_1, \ldots, e_m$ such that for every $1 \leq i \leq m$, we have $e_i \in E$, we define $\ell(\rho) = \ell(e_1) \cdot \ell(e_2) \cdots \ell(e_m)$. We then expend ℓ to multisets of paths. A *multiset* is a generalization of a set in which each element may appear more than once. We use multisets, as paths may have a flow greater than 1 flowing through them. Given a multiset of paths $P = \{\rho_1 \ldots, \rho_m\}$, we define $\ell(P) = \{\ell(\rho_1), \ldots, \ell(\rho_m)\}$. Note that $\ell(P)$ is a multiset of words in Σ^*.

Let $P = \{\rho_1, \ldots, \rho_m\}$ be a multiset of paths in $\langle E, V\rangle$, such that for every $1 \leq i \leq m$, we have that $\rho_i = e^i_1, \ldots, e^i_{k_i}$ is a path in $\langle E, V\rangle$ with $e^i_1 \in E_s$ and $e^i_{k_i} \in E^t$. Let $f \in \mathbb{N}^E$ be a flow function. We say that P is a *decomposition* of f if for every $e \in E$ it holds that $|\{(i, j) : 1 \leq i \leq m,\ 1 \leq j \leq k_i \text{ and } e^i_j = e\}| \leq f(e)$. That is, the number of times every edge $e \in E$ is traversed in paths in P is at most the flow through e. We say that P is *complete* if, additionally, it contains all paths reaching t. That is, for every $e \in E^t$ it holds that $|\{(i, j) : 1 \leq i \leq m,\ 1 \leq j \leq k_i \text{ and } e^i_j = e\}| = f(e)$. Since flow can get lost in the game, we do not require equality for all edges.

For a multiset M of words in Σ^*, we denote by M_L the restriction of M to words from L. That is, $M_L = \{w \in M : w \in L\}$. Then, for a decomposition P of f, the goal of Player 0 is to maximize the number of words in $\ell(P)$ that are also in L, namely, $|\ell(P)_L|$.

As Example 3 below demonstrates, a given the flow in the game may have several decompositions.

Example 3. See for example the RFG below.



Let f be a flow function described in the numbers labeling the edges (that is, the numbers labeling the edges in this example represent flow, rather than capacity). Let $L = a^* + b^*$ be the specification.

Let $\rho_1 = (s, a, v_1), (v_1, b, v_2), (v_2, a, t)$ and $\rho_2 = (s, b, v_1), (v_1, a, v_2), (v_2, b, t)$ be paths in the game. It is easy to verify that the multiset $P = \{\rho_1, \rho_1, \rho_2\}$ is a decomposition of the flow function f and that $\ell(P) = \{aba, aba, bab\}$. Thus, $\ell(P)_L = \emptyset$. Alternatively, let $\rho_3 = (s, a, v_1), (v_1, a, v_2), (v_2, a, t)$ and $\rho_4 = (s, b, v_1), (v_1, b, v_2), (v_2, b, t)$ be paths in the game. The multiset $P' = \{\rho_1, \rho_3, \rho_4\}$ is also a decomposition of f, with $\ell(P') = \{aba, aaa, bbb\}$. Thus, $\ell(P')_L = \{aaa, bbb\}$. In particular, we get that $|\ell(P)_L| \neq |\ell(P')_L|$. □

We define two variants of restricted flow games. One where we consider the most beneficial decomposition to Player 0, and one where we consider the least beneficial decomposition to Player 0. In both settings, Player 0 plays first and determines the policies for all her vertices. This models the fact that we want the setting to capture the most hostile environment when we refer to the flow itself.

In the first variant, which we call the maximal variant, the outcome of the game is given by

$$\mathit{outcome\_max}(\alpha, \beta) = \max\{|\ell(P)_L| : P \text{ is a decomposition of } f^{\alpha,\beta}\}.$$

We refer to a decomposition that achieves the maximum value as a maximal decomposition, and denote it (one of them, in case of multiplicity) by P_{max}. The maximal value of an RFG is then given by

$$\mathit{rval\_max}(\mathcal{G}) = \max_{\alpha \in F_0} \min_{\beta \in F_1} \mathit{outcome\_max}(\alpha, \beta).$$

That is, given strategies for both player, we take the decomposition that maximizes the number of words in the specification. This corresponds to a setting in which Player 0 plays first and determines the policies for all her vertices, then Player 1 responds by determining the policies for all her vertices, and finally, Player 0 chooses a decomposition that benefits her the most.

In the second variant, which we call the minimal variant, the outcome of the game is given by

$$\mathit{outcome\_min}(\alpha, \beta) = \min\{|\ell(P)_L| : P \text{ is a complete decomposition of } f^{\alpha,\beta}\}.$$

Since we are taking the minimal outcome, we require P to be a complete decomposition. Otherwise, we would have that for $P = \emptyset$, the outcome is always 0. We refer to a complete decomposition that achieves the minimal value as a minimal decomposition, and denote it by P_{min}. The minimal value of an RFG is then given by

$$\mathit{rval\_min}(\mathcal{G}) = \max_{\alpha \in F_0} \min_{\beta \in F_1} \mathit{outcome\_min}(\alpha, \beta).$$

This corresponds to a setting in which Player 0 plays first and determines the policies for all her vertices, and then Player 1 responds by determining the policies for all her vertices, and choosing a complete decomposition of the obtained flow in a way that benefits Player 0 the least.

3 Calculating the Outcome in Restricted Flow Games

In this section we study the complexity of calculating the outcome in RFGs. Recall that in the unrestricted settings, calculating the outcome of the game, given strategies for both players, can be done in polynomial time [27]. We show that in restricted flow games, calculating the maximal and minimal outcomes is NP-complete and co-NP-complete, respectively, and is independent of the branching mode of the specification automaton. On the other hand, determining whether the maximal outcome is positive depends on the representation of the specification, is NP-complete for specifications given by a UFA, and is NLOGSPACE-complete for ones given by a n NFA or DFA.

We consider the corresponding decision problems, which are parameterized by a threshold. Formally, for $\lambda \in \{U, N, D\}$, the input to a λ-*maximal outcome* problem, denoted λ-OUTCOME-MAX, is an RFG $\mathcal{G}$ with a specification given by a λ-auotmaton, a flow $f : E \to \mathbb{N}$ in $\mathcal{G}$, and a threshold $\gamma \in \mathbb{N}$. The goal is to decide whether there exists a decomposition P of f such that $|\ell(P)_L| \geq \gamma$. The input to a λ-*minimal outcome* problem (denoted λ-OUTCOME-MIN) is the same, but the goal is to decide whether for all complete decompositions of f it holds that $|\ell(P)_L| \geq \gamma$. Then, the input to the λ-*positive maximal outcome* and λ-*positive minimal outcome* problems, denoted λ-OUTCOME-MAX$^+$ and λ-OUTCOME-MIN$^+$, respectively, is similar, except that γ is not given, as it is fixed to 1. Thus, the goal in the max variant is to decide whether there exists a decomposition P of f such that $|\ell(P)_L| > 0$, and in the min variant to decide whether for all complete decompositions P of f, it holds that $|\ell(P)_L| > 0$.

3.1 Solving the maximal and minimal outcome problems

We start with the λ-maximal and λ-minimal outcome problems. For the lower bounds, we use reductions from the maximum 3-bounded 3-dimensional matching problem (3DM-3 problem, for short), shown to be APX-hard in [22]. That is, there exists a constant c such that it is NP-hard to find an approximation algorithm with approximation ratio better than c. In particular, it is NP-hard to solve the problem exactly. A similar reduction was shown in [24], from the 3DM-3 problem to the maximum restricted utilization problem of capacitated automata. In fact, we can show a simple reduction from the maximum restricted utilization problem to the D-maximal outcome problem, but we reduce directly from the 3DM-3 problem for completeness.

The input to the 3DM-3 problem is a set of triples $T \subseteq X \times Y \times Z$, where $|X| = |Y| = |Z| = n$. The number of occurrences of every element of $X \cup Y \cup Z$ in T is at most 3. The number of triples is $|T| \geq n$. The desired output is a 3-dimensional matching in T of maximal cardinality; i.e., a subset $T' \subseteq T$, such that every element of $X \cup Y \cup Z$ appears at most once in T', and $|T'|$ is maximal. The equivalent decision problem is parameterized by a threshold.

Theorem 1. *The λ-OUTCOME-MAX problem is NP-complete, for $\lambda \in \{U, N, D\}$.*

Proof. First, for every $\gamma \geq 1$, if $\mathit{outcome\_max}(\alpha, \beta) \geq \gamma$ then there is a decomposition P of f such that $|\ell(P)_L| \geq \gamma$. The decomposition P is a polynomial witness. Indeed, we can verify that P is a decomposition of f by counting the number of traversals of each edge in the graph. Calculating $\ell(P)$ requires one iteration over the edges in each path. We then count the number of words in $\ell(P)$ that are also in L. This can also be done in polynomial time for universal, nondeterministic and deterministic automata. Finally, the number of edges traversed in P is bounded by the total capacity of the game. Since capacities are given in unary, the total capacity is polynomial in the input, and so is the length of P.

We show the lower bound for the most restricted case, namely $\lambda = D$, by reducing from the 3DM-3 problem.

Given sets X, Y and Z with $|X| = |Y| = |Z| = n$, and a set of triples $T \subseteq X \times Y \times Z$, we construct an RFG $\mathcal{G} = \langle \Sigma, V_0, V_1, E, c, s, t, L \rangle$, as shown in Figure 1. First, $\Sigma = X \cup Y \cup Z$, $V_0 = \{v_0, v_1, v_2, v_3\}$, $V_1 = \emptyset$, $s = v_0$ and $t = v_3$. As for edges and capacities, there are n edges from each vertex to the next one, all with capacity 1. The edges from v_0 to v_1 correspond to elements in X, the edges from v_1 to v_2 correspond to elements in Y, and the edges from v_2 to v_3 correspond to elements in Z. Formally, $E = (\{v_0\} \times X \times \{v_1\}) \cup (\{v_1\} \times Y \times \{v_2\}) \cup (\{v_2\} \times Z \times \{v_3\})$, and $c \equiv 1$. We then define the flow in $\mathcal{G}$ to be $f \equiv 1$. That is, we have a flow of 1 in all edges.

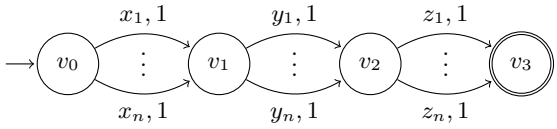


Fig. 1. The flow network in $\mathcal{G}$

To complete the reduction, let $L = \{x \cdot y \cdot z : (x, y, z) \in T\}$. Note that since the 3DM instance is 3-bounded, we have that $|L| = O(n)$. Thus, we can describe L by a DFA with $O(n)$ states.

We claim that for every $\gamma \geq 1$, the maximal matching T_{max} satisfies $|T_{max}| \geq \gamma$ iff the maximal decomposition P_{max} of f satisfies $|\ell(P_{max})_L| \geq \gamma$.

Every matching $T' \subseteq T$ corresponds to a set P of paths in $\mathcal{G}$. Since every element of $X \cup Y \cup Z$ appears at most once in a matching, every edge in the set of paths is traversed at most once. Thus, P is a decomposition of f. In addition, since $T' \subseteq T$, we have that for every $w \in \ell(P)$, it holds that $w \in L$. It follows that $|\ell(P)_L| = |P| = |T'|$.

Conversely, for decomposition P of f, denote by $P' \subseteq P$ the multiset of paths in P that corresponds to words in L. That is, $\ell(P)_L = \ell(P')_L = \ell(P')$. Let $T' \subseteq T$ be the subset of T corresponding to the paths in P'. Since P is a

decomposition of f, we know that every edge appears at most once in P, and in particular, in P'. Thus, T' is a matching such that $|T'| = |P'| = |\ell(P)_L|$.

Hence, $|T_{max}| \geq \gamma$ iff $|\ell(P_{max})_L| \geq \gamma$, and we are done. □

Theorem 2. *The λ-OUTCOME-MIN problem is co-NP-complete, for $\lambda \in \{U, N, D\}$.*

Proof. First, for every $\gamma \geq 1$, if $outcome\_min(\alpha, \beta) < \gamma$, then there is a decomposition P of f such that $|\ell(P)_L| < \gamma$. The decomposition P is a polynomial witness. As we have shown in the proof of Theorem 1, the size of P is polynomial in the input and we can verify it in polynomial time.

We show the lower bound for the most restricted case, namely $\lambda = D$. We modify the reduction from the 3DM-3 problem shown in Theorem 1, such that a maximal matching would induce a minimal complete decomposition. Thus, we reduce from the complement of the 3DM-3 problem.

Given an instance of the 3DM-3 problem, we construct a game $\mathcal{G}$ and flow f as in the proof of Theorem 1. We then define the specification to be $L' = \{x \cdot y \cdot z : (x, y, z) \notin T\}$. That is, the set of words corresponding to triples that are not in T. Recall the specification L described in the proof of Theorem 1, and consider $\bar{L}$, namely, the set of words that are not in L. We have that $L' = \bar{L} \cap (X \cdot Y \cdot Z)$. We can describe $\bar{L}$ by a DFA with $O(n)$ states, and $X \cdot Y \cdot Z$ by a DFA with $O(1)$ states. Thus, we can describe their intersection L' by a DFA with $O(n)$ states. Finally, we set $\gamma' = \max\{0, n - \gamma + 1\}$.

We claim that the maximal matching T_{max} satisfies $|T_{max}| < \gamma$ iff the minimal decomposition satisfies $\ell(P_{min})_{L'} \geq \gamma'$.

First, in case $\gamma > n$, we know that $|T| < \gamma$ for every matching T, in particular T_{max}. In addition, $\gamma > n$ implies that $\gamma' = 0$, and so $\ell(P)_{L'} \geq \gamma'$ for every decomposition P, in particular P_{min}. Thus, our reduction holds.

Assume now that $|T_{max}| < \gamma \leq n$, and let P be the corresponding decomposition of f. Namely, for every $(x, y, z) \in T_{max}$, P contains the path $(v_0, x, v_1), (v_1, y, v_2), (v_2, z, v_3)$. We note that P is not a complete decomposition, since its size is less than n. Consider some choice of paths P' such that $P \cup P'$ is a complete decomposition of f. There must exist such a choice, since the incoming and outgoing flow of all vertices is n. In addition, since T_{max} is a maximal matching, the paths in P' must correspond to triples that are not in T. Denote $P'' = P \cup P'$. Since T_{max} is maximal, we get that P'' is minimal. Finally, $|\ell(P'')_{L'}| = |\ell(P)_{L'}| + |\ell(P')_{L'}| = 0 + (n - |T_{max}|) > n - \gamma = \gamma' - 1$, and thus $|\ell(P'')_{L'}| \geq \gamma'$.

Conversely, assume that there is matching $T' \subseteq T$ such that $|T'| \geq \gamma$. Let P be the decomposition induced by T', and let P' be an expansion of P as above. Denote $P'' = P \cup P'$. We get that $|\ell(P'')_{L'}| = n - |T'| \leq n - \gamma < \gamma'$. Thus, $|\ell(P_{min})_{L'}| \leq |\ell(P'')_{L'}| < \gamma'$.

Hence, $|T_{max}| < \gamma$ iff $|\ell(P_{min})_{L'}| \geq \gamma'$, and we are done. □

3.2 Solving the positive maximal and minimal outcome problems

We now consider the problem of determining whether the maximal and minimal outcomes are positive, thus deciding the λ-positive maximal and minimal

outcome problems. We show that in some cases, depending on λ, solving the λ-OUTCOME-MAX$^+$ problem is easier than the general case. However, solving the λ-OUTCOME-MIN$^+$ problem is as hard as the general problem for all $\lambda \in \{U, N, D\}$..

Note that since every algorithm that approximates the solution of a problem within a multiplicative factor can determine whether the solution is positive or is 0, a lower bound for the complexity of solving the positivity problems is also a lower bound for all approximation algorithms.

We start with some simple constructions that we later use in our reductions.

Let $X = \{x_1, \ldots, x_n\}$ be a set of variables. We encode an assignment to X as a word $w \in \{T, F\}^n$ in the expected way. Thus, if $w = w_1 \cdots w_n$, then for every $1 \leq i \leq n$, the value of x_i is w_i. Consider a Boolean formula ψ over X. Let $L_\psi \subseteq \{T, F\}^n$ be the set of encodings of assignments to X that satisfy ψ. We say that ψ is in *conjunctive normal form*, CNF for short, if it is a conjunction of clauses, where a clause is a disjunction of literals. Alternatively, we say that ψ is in *disjunctive normal form*, DNF for short, if it is a disjunction of clauses, where a clause is a conjunction of literals. We show that when ψ is given in CNF or DNF, we can recognize L_ψ with a UFA or an NFA, respectively, whose number of states is polynomial in the size of ψ. We note that we prove this for a simple encoding over $\{T, F\}$. In further proofs, we might use an encoding over $\{T, F, \$\}$, where \$ is used as a delimiter. Modifying the proof to accommodate for the different encodings is simple.

Lemma 1. *Let ψ be a Boolean formula over $X = \{x_1, \ldots, x_n\}$. Let L_ψ be the set of encodings of assignments that satisfy ψ. Then*

1. *If ψ is given in DNF, there is an NFA of size $O(|\psi|)$ that recognizes L_ψ.*
2. *If ψ is given in CNF, there is a UFA of size $O(|\psi|)$ that recognizes L_ψ.*

Proof. First, assume that ψ is in DNF. Thus, there are clauses $c_1, \ldots, c_k$ such that $\psi = \bigvee_{i=1}^{k} c_i$, and each clause is a conjunction of literals.

We claim that we can recognize L_ψ with an NFA $\mathcal{A}_\psi$ with $k \cdot (n+1)$ states. The NFA $\mathcal{A}_\psi$ is the union of k DFAs, one for each clause in ψ. For every clause c_i, we construct a DFA with states $q_0, \ldots, q_n$, where q_0 is the initial state and q_n is the accepting state. For every $1 \leq j \leq n$, the state q_{j-1} corresponds to the variable x_j, and is connected to q_j. If x_j appears in c_i, there is a transition from q_{j-1} to q_j when reading T. If $\bar{x_j}$ appears in c_i, there is a transition from q_{j-1} to q_j when reading F. If neither appear, there is a transition from q_{j-1} to q_j when reading either T or F. The NFA $\mathcal{A}_\psi$ is the union of the DFAs, obtained by putting them side-by-side. Indeed, a word in their union encodes an assignment to X, such that at least in one of the clauses, all of the literals have positive value.

We assume now that ψ is in CNF. Thus, $\neg\psi$ is in DNF, and there is an NFA $\mathcal{A}$ of size $O(|\psi|)$ with $L(\mathcal{A}) = L_{\neg\psi}$. Let $\mathcal{A}'$ be the UFA obtained from $\mathcal{A}$ by setting $F' = Q \setminus F$. It is easy to verify that $L(\mathcal{A}') = \overline{L_{\neg\psi}}$. In addition, we know that $L_\psi = \{T, F\}^n \cap \overline{L_{\neg\psi}}$. We can recognize $\{T, F\}^n$ with a DFA of size $O(n)$.

It follows that we can recognize the intersection $\{T,F\}^n \cap \overline{L_{\neg\psi}}$ by putting $\mathcal{A}'$ and the DFA side-by-side. The obtained UFA is of size $O(|\psi|)$, and recognizes L_ψ. □

We now use this construction to prove NP-hardness of the U-OUTCOME-MAX$^+$ problem.

Theorem 3. *The U-OUTCOME-MAX$^+$ problem is NP-complete.*

Proof. Membership in NP follows from Theorem 1.

For the lower bound, we reduce from SAT. Let $X = \{x_1, \ldots, x_n\}$ be a set of variables, and ψ be a Boolean formula over X given in CNF. We encode an assignment to X as described above. We construct a game $\mathcal{G}$ as follows. First, $\Sigma = \{T, F\}$, $V_0 = \{v_i : 1 \leq i \leq n\} \cup \{t\}$, $V_1 = \emptyset$, and $s = v_1$. That is, we have $n+1$ vertices, one for each variable in X, and a target vertex. For each v_i, we have two outgoing edges, one labeled with T and one with F. The last variable vertex, v_n, is connected to the target vertex t. Formally,

$$E = \{(v_i, T, v_{i+1}), (v_i, F, v_{i+1}) : 1 \leq i \leq n-1\} \cup \{(v_n, T, t), (v_n, F, t)\}.$$

We define the capacity c and the flow f to be 1 in all edges. Finally, we define the specification L_ψ to be the set of all encodings of assignments to X that satisfy ψ. By Lemma 1, we can recognize L_ψ with a UFA of size $O(|\psi|)$.

We claim that there exists a decomposition P of f with $|\ell(P)_{L_\psi}| > 0$ iff ψ is satisfiable.

We identify a path in $\mathcal{G}$ with an assignment to X. Indeed, every path ρ_1 from v_0 to t must have $\ell(\rho_1) \in \{T,F\}^n$. If $|\ell(P)_{L_\psi}| > 0$, then there is a path $\rho_1 \in P$ such that $\ell(\rho_1) \in L_\psi$. Thus, the path ρ_1 encodes an assignment to X that satisfies ψ, and it is satisfiable.

Conversely, if ψ is satisfiable, then there is an assignment π to X that satisfies ψ. Let ρ_π be the path in $\mathcal{G}$ that corresponds to π. Hence, we have that $\ell(\rho_\pi) \in L_\psi$, and $|\ell(\{\rho_\pi\})_{L_\psi}| > 0$. □

On the other hand, for the cases $\lambda = N$ and $\lambda = D$, we can solve the positivity problem more efficiently.

Theorem 4. *The λ-OUTCOME-MAX$^+$ problem is NLOGSPACE-complete, for $\lambda \in \{N, D\}$.*

Proof. Let $\mathcal{G} = \langle \Sigma, V_0, V_1, E, c, s, t, L \rangle$ be an RFG, f be a flow, and $\mathcal{A}$ be the DFA or NFA for which $L = L(\mathcal{A})$. The maximal decomposition P of f satisfies $|\ell(P)_L| > 0$ iff there is a path from s to t traversing edges with positive flow in f, such that there is an accepting run of $\mathcal{A}$ on the word labeling the path. Indeed, if there is such a path ρ, then $P' = \{\rho\}$ is a decomposition of f with $|\ell(P')_L| = 1$. On the other hand, if $|\ell(P)_L| > 0$, then there is a path $\rho \in P$ such that $\ell(\rho) \in L(\mathcal{A})$, and so there is an accepting run of $\mathcal{A}$ on $\ell(\rho)$. Thus, we can guess a path from s in $\mathcal{G}$ and a run of $\mathcal{A}$, one transition at a time on the fly, such that at each step, the edge guessed in $\mathcal{G}$ and the transition guessed

in $\mathcal{A}$ are labeled with the same letter. In addition, we make sure every edge guessed in $\mathcal{G}$ has a positive flow through it. Hence, the problem can be solved in NLOGSPACE.

For the lower bound, we reduce from the reachability problem, known to be NLOGSPACE-hard [29]. Given a DAG $G = \langle V, E \rangle$ and vertices $s, t \in V$, we let Player 0 control all vertices, label all edges with some letter a, let the capacity and flow in all edges be 1, and set the specification to be $L = a^*$. If there is a path from s to t, then it must be labeled with a word in L. Thus, the multiset P containing this path is a decomposition of the flow with $|\ell(P)|_L > 0$. Conversely, if there is such a decomposition, there must be a path from s to t, and t is reachable from s. □

In the minimal variant, solving the positivity problem remains as hard as the general problem for all types of specifications.

Theorem 5. *The λ-OUTCOME-MIN$^+$ problem is co-NP-complete, for $\lambda \in \{U, N, D\}$.*

Proof. The upper bound follows from Theorem 2.

For the lower bound, consider the reduction shown in the proof of Theorem 2. The case where $\gamma = n$ is the *perfect* variant. That is, the problem of determining whether there is a matching that matches all elements in $X \cup Y \cup Z$. This was shown to be NP-complete in the bounded variant of 3-dimensional matching in [30]. In addition, when $\gamma = n$, we get that $\gamma' = 1$. Thus, we get that the reduction is from the complement of the perfect variant to the positivity variant, and achieves the desired lower bound. □

4 Calculating the Value in Restricted Flow Games

In this section we study the problem of finding the maximal and minimal values of RFGs. We show that the difference in the complexity of calculating the maximal and minimal outcomes translates to a difference in the complexity of calculating the value. Thus, the maximal version has one more quantifier alternation, which results in a "step up" in the polynomial hierarchy. The polynomial hierarchy is a hierarchy of complexity classes that use oracles. The lowest level is defined $\Sigma_0^P = \Pi_0^P = P$. Then, we define $\Sigma_{i+1}^P = \text{NP}^{\Sigma_i^P}$ and $\Pi_{i+1}^P = \text{co-NP}^{\Pi_i^P}$, where A^B is the set of decision problems that can be solved in A using an oracle to some problem complete in class B. Thus, Σ_2^P is the set of problems that can be solved using a polynomial time nondeterministic Turing Machine with an oracle to some NP-complete problem.

We consider the decision problems, which are parameterized by a threshold. Formally, the input to a *maximum λ-restricted flow game problem*, λ-RFG-MAX problem, for short, is an RFG $\mathcal{G}$ and a threshold $\gamma \in \mathbb{N}$. The goal is to decide whether $\mathit{rval\_max}(\mathcal{G}) \geq \gamma$. The input to the *minimum λ-restricted flow game problem*, λ-RFG-MIN problem, for short, is the same, and the goal is to decide whether $\mathit{rval\_min}(\mathcal{G}) \geq \gamma$. As in the problem of calculating the maximal

and minimal outcomes, we also study the special case with $\gamma = 1$, namely the positivity problems. Formally, the input to the *positive maximum* and *positive minimum* λ*-restricted flow game problems* (λ-RFG-MAX$^+$ and λ-RFG-MIN$^+$, for short) is an RFG $\mathcal{G}$, and the goal is to decide whether $rval\_max(\mathcal{G}) > 0$ and whether $rval\_min(\mathcal{G}) > 0$, respectively.

4.1 Solving the maximum and minimum restricted flow problems

Recall that in the unrestricted case, the FG problem is in Σ_2^P. Since calculating the maximal outcome is in NP, we can "step up" in the hierarchy by using an oracle.

Theorem 6. *The λ-RFG-MAX problem is in Σ_3^P, for $\lambda \in \{U, N, D\}$.*

Proof. Consider an RFG and a threshold $\gamma \in \mathbb{N}$. Given a strategy α for Player 0, deciding whether there is a strategy β for Player 1 such that $outcome\_max(\alpha, \beta) < \gamma$ can be done by guessing β in NP, and using a co-NP oracle for deciding whether $outcome\_max(\alpha, \beta) < \gamma$ (as we have shown in Theorem 1). Thus, it is in Σ_2^P. Consequently, deciding whether there is a strategy α for Player 0 such that $outcome\_max(\alpha, \beta) \geq \gamma$ for every $\beta \in F_1$ can be done in NP with a Σ_2^P oracle, by guessing α and checking it as above. It follows that the λ-RFG-MAX problem is in Σ_3^P. □

We now show that this "step up" in the polynomial hierarchy is necessary by proving a matching lower bound. To show that, we use a reduction from QBF$_3$.

For $k \geq 1$, the problem QBF$_k$ is the satisfiability problem of quantified Boolean formulas with $k-1$ alternations of quantifiers, where the most external quantifier is "exists". By [31], the problem QBF$_k$ is Σ_k^P-complete when k is odd and the formula is given in CNF, and when k is even and the formula is given in DNF.

For the QBF$_3$ problem, let $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, and $Z = \{z_1, \ldots, z_k\}$ be sets of variables. We denote by $\bar{X}$, $\bar{Y}$ and $\bar{Z}$ the sets of negative literals. We also denote $W = X \cup Y \cup Z$ and $\bar{W} = \bar{X} \cup \bar{Y} \cup \bar{Z}$. Let ψ be a Boolean propositional formula over the variables in W, and let $\theta = \exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m \exists z_1 \ldots \exists z_k \psi$.

We assume that ψ is given in CNF. Thus, $\psi = \bigwedge_{i=1}^{r} c_i$ for clauses $c_1, \ldots, c_r$.

Let π be an assignment to $W' \subseteq W$. We denote by $\psi|_\pi$ the Boolean formula obtained from ψ by substituting all literals from $W' \cup \bar{W'}$ with their value in π. The obtained formula $\psi|_\pi$ is a Boolean formula over $W \setminus W'$. We then denote by $\theta|_\pi$ the quantified Boolean formula obtained from θ by replacing ψ with $\psi|_\pi$ and deleting the variables in W' from the prefix of quantifiers in θ. The obtained formula $\theta|_\pi$ is a quantified Boolean formula over $W \setminus W'$. For example, if $W' = X$, we get that $\theta|_\pi = \forall y_1 \ldots \forall y_m \exists z_1 \ldots \exists z_k \psi|_\pi$.

We say that π satisfies θ if $\theta|_\pi$ holds. Thus, we get that θ holds iff there exists an assignment π to X such that π satisfies θ. Equivalently, $\theta|_\pi$ holds iff for all assignments π' to Y we have that π' satisfies $\theta|_\pi$.

Theorem 7. *The U-RFG-MAX problem is Σ_3^P-hard.*

Proof. We show that the U-RFG-MAX problem is Σ_3^P-hard using a reduction from QBF_3. Given a formula θ as above, we construct an RFG $\mathcal{G}_\theta$ such that $rval\_max(\mathcal{G}_\theta) \geq 1$ iff θ holds. Thus, we set $\gamma = 1$.

First, we encode an assignment to the variables in W as a word $w = w_z \cdot \$ \cdot w_x \cdot w_y \in \{T, F, \$\}^*$, where $w_x, w_y, w_z \in \{T, F\}^*$ encode assignments to X, Y, and Z, respectively, as we described in Lemma 1.

We define the game $\mathcal{G}_\theta$ so that the choices of Player 0 induce w_x, the choices of Player 1 induce w_y, and w_z is chosen when determining the outcome.

As can be seen in Figure 2, the game consists of a single vertex for every variable in W, as well as a vertex for reading \$, and a target vertex t. We refer to vertices by the variable associated with them.

The source vertex is z_1. Every vertex is then connected to the vertex of the next variable with two edges, one labeled T and one labeled F, both with capacity 1. We connect z_n to the \$ vertex, and then connect it to x_1 with a \$-labeled edge with capacity 1. Finally, x_n is connected to y_1, and y_m is connected to the target vertex.

We let Player 0 control the vertices associated with variables in X and Z, and Player 1 control the vertices associated with variables in Y. Since the vertex associated with \$ has a single outgoing edge, we arbitrarily let Player 0 control it.

The \$-labeled edge is a minimum cut of the graph, assuring that only a single unit of flow reaches the X and Y vertices. Thus, a strategy for Player 1 is equivalent to choosing either T or F for each variable in Y. Similarly, a strategy for Player 0 induces an assignment for variables in X. Player 0 can also determine the flow leaving the source vertex. However, saturating the two edges leaving the source vertex is a dominant strategy for Player 0, and so we can consider only the case where the outgoing edges of Z vertices are saturated.

Finally, we define the specification L_ψ to be the set of assignments that satisfy ψ. The specification can be recognized by a UFA of size $O(|\psi|)$, as described in Lemma 1.

We claim that θ holds iff $rval\_max(\mathcal{G}_\theta) \geq 1$.

First, assume that θ holds, and let π_X be an assignment to X that satisfies it. Let $\alpha_\pi \in F_0$ be the strategy that corresponds to π_X. That is, for each $1 \leq i \leq n$, the strategy α_π directs the single unit of flow entering the x_i vertex to the T-labeled edge if $\pi_X(x_i) = 1$, and directs it to the F-labeled edge if $\pi_X(x_i) = 0$. In order to show that α_π ensures that the outcome of the game is 1, consider a strategy $\beta \in F_1$ for Player 1. Let π_β be the assignment to Y that corresponds to β. That is, for every $1 \leq j \leq m$, we have $\pi_\beta(y_j) = 1$ iff β directs the single unit of flow entering y_j to the T-labeled edge. We know that $\theta|_{\pi_X}$ holds. Thus, every assignment π' to Y satisfies $\theta|_{\pi_X}$. In particular, $\theta|_{\pi_X, \pi_\beta}$ holds. It follows that there exists an assignment π_Z to Z that satisfies $\psi|_{\pi_X, \pi_\beta}$.

Let $w = w_z \cdot \$ \cdot w_x \cdot w_y$ be the word that encodes the assignment π_X, π_β and π_Z. It follows that $w \in L_\psi$. In addition, the flow in the game can be decomposed

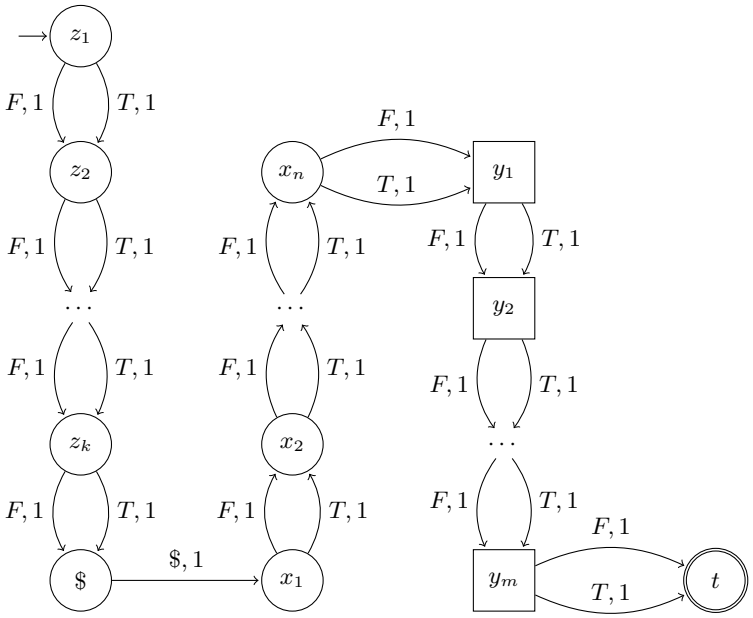


Fig. 2. The flow network in $\mathcal{G}_\theta$

such that the path that corresponds to w is in the decomposition. Thus, we get that $\mathit{outcome\_max}(\alpha_\pi, \beta) = 1$, as desired.

Assume now that $\mathit{rval\_max}(\mathcal{G}_\theta) \geq 1$. Thus, there is a strategy $\alpha \in F_0$ for Player 0 such that for every strategy $\beta \in F_1$, it holds that $\mathit{outcome\_max}(\alpha, \beta) \geq 1$. Let π_α be the assignment to X that corresponds to α. That is, for every $1 \leq i \leq n$, we have that $\pi_\alpha(x_i) = 1$ iff α directs the flow entering x_i to the T-labeled edge. We claim that π_α satisfies θ. Let π_Y be some assignment to variables in Y, and $\beta_\pi \in F_1$ be the strategy that corresponds to π_Y. That is, for all $1 \leq j \leq m$, the strategy β_π directs the flow entering y_j to the T-labeled edge iff $\pi_Y(y_j) = 1$. It follows that $\mathit{outcome\_max}(\alpha, \beta_\pi) \geq 1$. Thus, there is a decomposition P of f^{α,β_π} such that $|\ell(P)_{L_\psi}| \geq 1$. In particular, there is a path in the decomposition that encodes an assignment that satisfies ψ. This assignment must agree with π_α and π_Y. Thus, there must be an assignment to Z that satisfies $\psi|_{\pi_\alpha, \pi_Y}$, and θ holds. □

Note that the specification L_ψ used in the proof of Theorem 7 cannot be recognized by a deterministic automaton with a polynomial number of states. The richness of the universal model allowed us to simply let the players induce an assignment, and check if it gives a true value to θ when calculating the outcome. In the deterministic case, we need a more complex game that "helps" in checking the truth value of θ.

Theorem 8. *The λ-RFG-MAX problem is Σ_3^P-hard, for $\lambda \in \{N, D\}$.*

Proof. We show the lower bound for the more restricted case, $\lambda = D$, by reducing from QBF_3.

Let θ and ψ be as above. We assume that there is $d \geq 1$ such that every literal in $W \cup \bar{W}$ appears in ψ exactly d times. It is easy to see that every formula in CNF can be converted with at most a quadratic blow-up to an equivalent formula that satisfies this condition.

We construct an RFG $\mathcal{G}_\theta$ such that the maximal flow in the game is $(k+1) \cdot r$, and that it can be decomposed into $(k+1) \cdot r$ words in L iff θ holds. Thus, we set $\gamma = (k+1) \cdot r$, and so $\mathit{rval\_max}(\mathcal{G}_\theta) \geq \gamma$ iff θ holds.

The construction is a combination of three constructions: the reduction from 3SAT to the 3-dimensional matching problem, shown in [16] (Thm. 3.2), the reduction from QBF_2 to the flow game problem, shown in [27] (Thm. 2), and the reduction shown here in Theorem 1.

The graph includes three parts: one for determining the values of variables in X and Y, one for ensuring that all clauses are satisfied, and one for ensuring that variables in Z have legal values. In addition, the specification L ensures that all clauses are satisfied legally and that the values of variables in Z are consistent.

First, we construct *variable vertices* for variables in $X \cup Y$, as shown in Figure 3. Vertices associated with variables in X are in V_0, and the vertices associated with variables in Y are in V_1. There is an edge from the source $s \in V_0$ to each variable vertex, with capacity $2d$ and labeled with x or y, respectively.

Each variable vertex is then connected to two *literal vertices*, each associated with choosing a positive or a negative value to this variable. The edges have capacity $2d$ and are labeled with $*$. Literal vertices associated with literals in $X \cup \bar{X}$ are in V_1, and have an outgoing edge to a sink, with capacity d. Literal vertices associated with literals in $Y \cup \bar{Y}$ are in V_0.

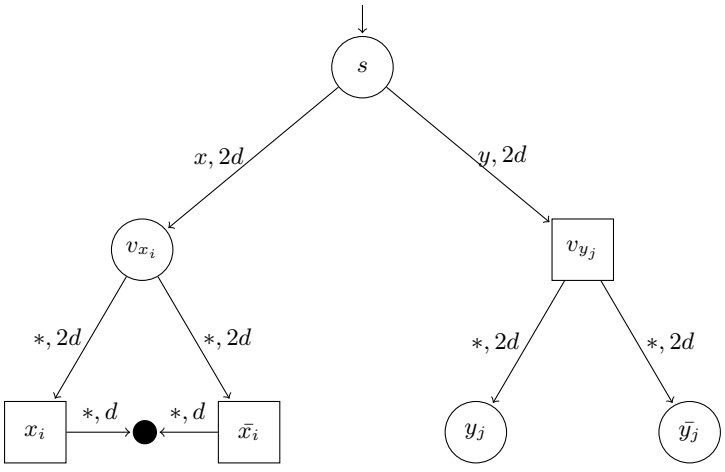


Fig. 3. The variable and literal vertices in $\mathcal{G}_\theta$, where $1 \leq i \leq n$ and $1 \leq j \leq m$.

We now define a layer of *clause vertices*. Each clause c_j, for $1 \leq j \leq r$, has a single vertex v_{c_j}, controlled by Player 0. Each clause vertex is connected to the target t with a single edge with capacity 1, labeled with c_j. In addition, each literal vertex is connected to each vertex associated with a clause that contains the literal. Each edge has capacity 1 and is labeled with $*$.

Finally, we define two vertices v_z and v_t for handling variables in Z, as shown in Figure 4. Both vertices are in V_0. The vertex v_z is used to choose the value of the variables in Z. There are $2kr$ edges from the source s to v_z, one for each literal and clause pair. That is, for every $1 \leq i \leq k$ and $1 \leq j \leq r$, we have the edges $(s, z_{i,j}, v_z)$ and $(s, \bar{z_{i,j}}, v_z)$. Each edge has capacity 1. Then, we have an edge from v_z to each clause vertex. Thus, for $1 \leq j \leq r$, we add an edge (v_z, c_j, v_{c_j}) with capacity 1.

The vertex v_t is then used to ensure the "truthfulness" of the values chosen for variables in Z. We connect v_z to v_t and v_t to the target t, in such a way that we could find a flow of words from the specification L iff the values given to variables in Z are consistent. We connect v_z to v_t with an edge for each variable in Z and each clause with capacity 1. That is, for every $1 \leq i \leq k$ and $1 \leq j \leq r$ we have the edges $(v_z, a_{i,j}, v_t)$ and $(v_t, b_{i,j}, t)$, both with capacity 1.

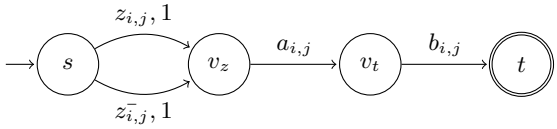


Fig. 4. The vertices v_z and v_t, where $1 \leq i \leq k$ and $1 \leq j \leq r$.

It is easy to verify that the size of the graph and the total capacity are polynomial in the size of ψ.

We now need to define the specification. Note that the alphabet we used is $\Sigma = \{*, x, y\} \cup \{c_j\}_{1 \leq j \leq r} \cup \{z_{i,j}, \bar{z}_{i,j}, a_{i,j}, b_{i,j}\}_{1 \leq i \leq k, 1 \leq j \leq r}$. This is also polynomial in the size of the input.

First, flow coming from a clause vertex into t originates in a literal that appears in that clause. For variables in $X \cup Y$, this follows from the construction. Thus, we allow all flow that goes through these variable vertices, and define $L_{x,y} = x\Sigma^* \cup y\Sigma^*$.

In Z, every literal induces an edge from s to v_z. A flow from v_z to a clause vertex is legal iff it came from a literal that appears in that clause. Thus, we define $L_{z,c} = \{z_{i,j} c_j c_j\}_{z_i \in c_j} \cup \{\bar{z_{i,j}} c_j c_j\}_{\bar{z_i} \in c_j}$.

Flow coming from v_t to t is then used to ensure the consistency of the values chosen for variables in Z. Consider a variable $z_i \in Z$. There are r incoming edges to t that are associated with z_i, one for each clause. We direct flow through these edges that comes from the edge associated with the value of z_i that was not chosen. Thus, if z_i is true, the flow goes through edges labeled with $\bar{z_{i,j}}$. We ensure

that only one value can be given to z_i by "intertwining" the legal flows for true and false values. Thus, we define $L_{z,t} = \{z_{i,j}a_{i,j}b_{i,j}, z_{i,j}^{-}a_{i,j+1}b_{i,j}\}_{1\leq i\leq k, 1\leq j\leq r}$, where $j+1$ is calculated modulo r.

Finally, we define $L = L_{x,y} \cup L_{z,c} \cup L_{z,t}$. It is easy to verify that L can be recognized by a DFA of size polynomial in $|\theta|$.

First, we note that the only vertices where the strategies affect the flow are the variable vertices. Indeed, the incoming flow to every other vertex in the graph is greater than its outgoing capacity. Since saturating all outgoing edges from s is a dominant strategy for Player 0, it follows that all edges, besides those between variable and literal vertices, are saturated, regardless of the strategies. Thus, Player 0 and Player 1 have control over directing $2d$ units of flow to the positive or negative literal vertices associated with each variable in V_0 or V_1, respectively. We also note that for both players, directing all $2d$ units of flow to the same literal vertex is a dominant strategy, and so we can consider just these strategies.

We now prove the correctness of the reduction, thus that $\mathit{rval\_max}(\mathcal{G}_\theta) \geq \gamma$ iff θ holds.

Assume θ holds. Let π_x be an assignment to X that satisfies θ. Let $\alpha_\pi \in F_0$ be the strategy that directs the incoming flow in each variable vertex in V_0 according to π. That is, if $\pi_x(x_i) = 1$, the strategy α_π directs $2d$ units of flow to the positive literal vertex of x_i, and if $\pi_x(x_i) = 0$, it directs $2d$ units of flow to the negative literal vertex of x_i. We show that for every $\beta \in F_1$, it holds that $\mathit{outcome\_max}(\alpha, \beta) = (k+1)\cdot r$. Directing flow to both literal vertices is a dominated strategy for Player 1. Thus, it is enough to show that Player 0 can guarantee an outcome of $(k+1)\cdot r$ against a strategy $\beta \in F_1$, such that it directs all incoming flow to a variable vertex in V_1 to only one literal vertex.

Denote by π_β the assignment to Y that assigns values according to β. That is, $\pi_\beta(y_i) = 1$ iff the strategy β directs $2d$ units of flow to the positive literal vertex. Since π_X satisfies θ, there exists an assignment π_Z to Z such that it satisfies $\psi|_{\pi_X,\pi_\beta}$. Denote by π the assignment to W that agrees with π_X, π_β and π_Z. We know that π satisfies ψ.

We construct a decomposition P of $f^{\alpha_\pi,\beta}$, such that $|\ell(P)_L| = (k+1)\cdot r$. Thus, $\mathit{outcome\_max}(\alpha_\pi, \beta) \geq (k+1)\cdot r$.

For every clause c_j, for $1 \leq j \leq r$, there is a literal φ_j that is assigned a positive value in π. Assume $\varphi_j = x_i$ for some $1 \leq i \leq n$. Denote by ρ_{c_j} the path $(s, x, v_{x_i}), (v_{x_i}, *, x_i), (x_i, *, c_j), (c_j, c_j, t)$. That is, the path that traverses the variable vertex, literal vertex and clause vertex that correspond to x_i and c_j. The edge $(x_i, *, c_j)$ exists because c_j contains x_i. Note that $\ell(\rho_{c_j}) = x**c_j \in L_{x,y}$. When φ_j is in $\bar{X} \cup Y \cup \bar{Y}$, we define ρ_{c_j} in a similar way. Now, assume $\varphi_j = z_i$ for some $1 \leq i \leq k$. In this case, we define ρ_{c_j} to be the path $(s, z_{i,j}, v_z), (v_z, c_j, c_j), (c_j, c_j, t)$. We have that $\ell(\rho_{c_j}) = z_{i,j}c_jc_j$. Since z_i appears in c_j, we have that $\ell(\rho_{c_j}) \in L_{z,c}$. When $\varphi_j \in \bar{Z}$, we define ρ_{c_j} in a similar way. Let $P_c = \{\rho_{c_j}\}_{1\leq j\leq r}$.

Consider indices $1 \leq i \leq k$ and $1 \leq j \leq r$. If $\pi(z_i) = 1$, let $\rho_{i,j}$ be the path $(s, z_{i,j}^{-}, v_z)$, $(v_z, a_{i,j+1}, v_t)$, $(v_t, b_{i,j}, t)$ (with $j+1$ being calculated modulo r). If

$\pi(z_i) = 0$, let $\rho_{i,j} = (s, z_{i,j}, v_z), (v_z, a_{i,j}, v_t), (v_t, b_{i,j}, t)$. In both cases, it is easy to verify that $\ell(\rho_{i,j}) \in L_{z,t}$. Let $P_z = \{\rho_{i,j}\}_{1 \leq i \leq k, 1 \leq j \leq r}$. Finally, let $P = P_c \cup P_z$. As noted above, all paths in P correspond to words in L, and so $|\ell(P)_L| = |P| = (k+1) \cdot r$. In addition, it is easy to verify that P is a decomposition of $f^{\alpha_\pi, \beta}$. Thus, $\mathit{outcome\_max}(\alpha_\pi, \beta) = (k+1) \cdot r$. That is, the strategy α_π for Player 0 ensures an outcome of at least γ, and so $\mathit{rval\_max}(\mathcal{G}_\theta) \geq \gamma$.

For the other direction, assume that $\mathit{rval\_max}(\mathcal{G}_\theta) \geq \gamma$. Let $\alpha \in F_0$ be a strategy for Player 0 that ensures an outcome of at least γ. Strategies that direct all incoming flow to a variables vertex to only one literal vertex are dominant strategies for Player 0, and so we can assume α is such a strategy. Denote by π_α the assignment to X that assigns values according to how α directs flow entering variable vertices. That is, $\pi_\alpha(x_i) = 1$ if the strategy α directs the $2d$ units of flow entering the variable vertex of x_i to the positive literal vertex, and $\pi_\alpha(x_i) = 0$ if α directs the flow to the negative literal vertex.

We claim that π_α satisfies θ. To show that, let π_Y be an assignment to Y. We need to show that there is an assignment to Z that satisfies $\psi|_{\pi_\alpha, \pi_Y}$.

Let $\beta_\pi \in F_1$ be the strategy that directs flow entering variable vertices according to π_Y. That is, if $\pi_Y(y_i) = 1$, the strategy β_π directs $2d$ units of flow to the positive literal vertex of y_i, and if $\pi_Y(y_i) = 0$, it directs $2d$ units of flow to the negative literal vertex of y_i. By the way we chose α, we know that $\mathit{outcome\_max}(\alpha, \beta_\pi) \geq (k+1) \cdot r$. Thus, there is a decomposition P of f^{α, β_π} such that $|\ell(P)_L| \geq (k+1) \cdot r$. Since the incoming capacity of t is exactly $(k+1) \cdot r$, we know that $|P| = |\ell(P)_L| = (k+1) \cdot r$. In particular, we know that there are $k \cdot r$ paths in P that traverse the edges between v_t and t.

Consider $1 \leq i \leq k$. For all $1 \leq j \leq r$, denote by $\rho^i_j \in P$ the path that traverses the edge $(v_t, b_{i,j}, t)$. Since $\ell(\rho^i_j) \in L$, we must have $\ell(\rho^i_j) = z_{i,j} a_{i,j} b_{i,j}$ or $\ell(\rho^i_j) = z^-_{i,j} a_{i+1,j} b_{i,j}$. We claim that either $\ell(\rho^i_j) = z_{i,j} a_{i,j} b_{i,j}$ holds for all j, or $\ell(\rho^i_j) = z^-_{i,j} a_{i,j+1} b_{i,j}$ holds for all j. Indeed, if $\ell(\rho^i_1) = z^-_{i,1} a_{i,2} b_{i,1}$, then it cannot be that $\ell(\rho^i_2) = z_{i,2} a_{i,2} b_{i,2}$, because the edge $(v_z, a_{i,2}, v_t)$ has capacity 1. Thus, $\ell(\rho^i_2) = z^-_{i,2} a_{i,2} b_{i,2}$. Inductively, we get that all ρ^i_j must traverse $(s, z^-_{i,j}, v_z)$. On the other hand, if $\ell(\rho^i_1) = z_{i,1} a_{i,1} b_{i,1}$, then it cannot be that $\ell(\rho^i_r) = z^-_{i,r} a_{i,1} b_{i,r}$. Thus, $\ell(\rho^i_r) = z_{i,r} a_{i,r} b_{i,r}$. We continue similarly to get that all ρ^i_j must traverse $(s, z_{i,j}, v_z)$. We let π_P be the assignment to Z that chooses the negation of ρ^i_j. That is, if all ρ^i_j traverse $(s, z^-_{i,j}, v_z)$ then $\pi_P(z_i) = 1$. If all ρ^i_j traverse $(s, z_{i,j}, v_z)$, then $\pi_P(z_i) = 0$.

We claim that π_P satisfies $\psi|_{\pi_\alpha, \pi_Y}$. To show this, we consider the r units of flow that reach t from the clause vertices. Let $1 \leq j \leq r$, and let $\rho_j \in P$ be the path that traverses the edge (v_{c_j}, c_j, t). By the structure of the game, we know that ρ_j goes through a literal vertex or through v_z. Assume that ρ_j goes through the literal vertex x_i. Then, $\rho_j = (s, x, v_{x_i}), (v_{x_i}, *, x_i), (x_i, *, v_{c_j}), (v_{c_j}, c_j, t)$. It follows that α directs the flow entering v_{x_i} to x_i, and thus, $\pi_\alpha(x_i) = 1$. In addition, since the edge $(x_i, *, c_j)$ exists in the graph, we know that x_i appears in c_j. So c_j has a positive value in $\psi|_{\pi_\alpha, \pi_Y}$. We use similar reasoning in the case ρ_j goes through a literal vertex in $\bar{X} \cup Y \cup \bar{Y}$. Now, assume ρ_j goes through v_z. Thus, ρ_j traverses the vertices s, v_z, v_{c_j} and t. Since $\ell(\rho_j) \in L$, we know

that $\ell(\rho_j) = z_{i,j}c_jc_j$ and z_i appears in c_j, or $\ell(\rho_j) = z^-_{i,j}c_jc_j$ and $\bar{z}_i$ appears in c_j. We assume w.l.o.g that the first case is true. Thus, ρ_j traverses the edge $e = (s, z_{i,j}, v_z)$. Since the capacity of e is 1, it follows that ρ^i_j does not traverse e. Hence, it traverses $(s, z^-_{i,j}, v_z)$, and $\pi_P(z_i) = 1$. Since z_i appears in c_j and has a positive value, we get that c_j is satisfied by π_P. Thus, all clauses are satisfied by π_α, π_Y and π_P, and π_P satisfies $\psi|_{\pi_\alpha,\pi_Y}$. It follows that π_α satisfies θ, and θ holds. □

We can now conclude with the tight complexity for the problem:

Theorem 9. *The λ-RFG-MAX problem is Σ^P_3-complete, for $\lambda \in \{U, N, D\}$.*

We continue to minimal variant. We can view this variant as the problem of deciding whether the exists a strategy α for Player 0, such that for every strategy β for Player 1 and *every* complete decomposition P of $f^{\alpha,\beta}$, we have that $|\ell(P)_L| \geq \gamma$. In this case, there is only one quantifier alternation. Thus, there is no increase in the number of alternation with respect to the unrestricted settings, and we remain in the same level of the polynomial hierarchy.

We start by showing a simple construction that we use in several proofs.

Lemma 2. *Given an unrestricted flow game $\mathcal{G}$, there is an RFG $\mathcal{G}'$ such that $value(\mathcal{G}) = rval\_max(\mathcal{G}') = rval\_min(\mathcal{G}')$.*

Proof. Given an unrestricted flow game $\mathcal{G}$, we obtain $\mathcal{G}'$ by labeling all edges with some letter a, and then defining $L = a^*$. The specification L can be recognized by a single state DFA. Since all paths in the game correspond to words in L, the maximal and minimal decompositions of a flow f coincide, and the size of their restriction to L must be equal to the value of f. Thus, trying to maximize the flow in $\mathcal{G}$ coincides with trying to maximize the maximal and minimal outcomes of $\mathcal{G}'$. □

Theorem 10. *The λ-RFG-MIN problem is Σ^P_2-complete, for $\lambda \in \{U, N, D\}$.*

Proof. Consider an RFG $\mathcal{G}$ and a threshold $\gamma \in \mathbb{N}$. Given a strategy α for Player 0, deciding whether there exists a strategy β for Player 1 such that $outcome\_min(\alpha, \beta) < \gamma$ can be done in NP, by guessing β and a complete decomposition P of $f^{\alpha,\beta}$ such that $|\ell(P)_L| < \gamma$. Calculating $f^{\alpha,\beta}$, verifying that P is a complete decomposition of $f^{\alpha,\beta}$ and computing $\ell(P)$ can all be done in polynomial time. Calculating $|\ell(P)_L|$ requires checking the membership of words in L, which can be done in polynomial time for all λ. Consequently, deciding whether there is a strategy α for Player 0 such that $outcome\_min(\alpha, \beta) \geq \gamma$ can be done in NP with a co-NP oracle, by guessing α and checking it as above. Thus, we have shown membership in Σ^P_2.

For the lower bound, we reduce from the unrestricted flow game problem, shown to be Σ^P_2-hard in [27]. Given an unrestricted flow game $\mathcal{G}$, the reduction outputs $\mathcal{G}'$ as described in Lemma 2. Correctness is immediate. □

4.2 Solving the positive maximum and minimum restricted flow problems

We continue to the positivity problems and show that their complexity is strongly related to the complexity of calculating the maximal and minimal outcomes: In the cases where the problem of determining whether the maximal outcome is positive is easier than the general problem, determining whether the maximal value is positive is also easier than the general problem. In other cases, the complexity of determining whether the maximal and minimal values are positive coincides with the complexity of the general problem.

Theorem 11. *The U-RFG-MAX$^+$ problem is Σ_3^P-complete.*

Proof. The reduction used for proving the Σ_3^P lower bound in the proof of Theorem 7 is such that the maximal value of the RFG is 1 if θ holds and is 0 otherwise. Hence, the reduction works for the positivity problem as well. Thus, both the upper and lower bounds follow from Theorem 7. □

Theorem 12. *The N-RFG-MAX$^+$ and D-RFG-MAX$^+$ problems are Σ_2^P-complete.*

Proof. For the upper bound, let $\alpha \in F_0$ and $\beta \in F_1$ be strategies for Player 0 and Player 1, respectively. We can calculate the flow in the game $f^{\alpha,\beta}$ in polynomial time. Checking if there is a decomposition P of $f^{\alpha,\beta}$ such that $\ell(P)_L > 0$ can be done in NLOGSPACE, as shown in Theorem 4. It follows that given $\alpha \in F_0$, the problem of checking if there is a $\beta \in F_1$ such that $\mathit{outcome\_max}(\alpha, \beta) = 0$ is in NP, by guessing β. Consequently, deciding whether there is $\alpha \in F_0$ such that for every $\beta \in F_1$ we have $\mathit{outcome\_max}(\alpha, \beta) > 0$, can be done by a nondeterministic polynomial-time Turing machine with an NP oracle.

For the lower bound, we reduce from the problem of approximating the value of an unrestricted flow game within any multiplicative factor, shown to be Σ_2^P-hard in [27]. Given an unrestricted flow game $\mathcal{G}$, the reduction outputs $\mathcal{G}'$ as described Lemma 2. Since every approximation algorithm can determine whether the outcome is positive, the correctness is immediate. □

Theorem 13. *The λ-RFG-MIN$^+$ problem is Σ_2^P-complete, for $\lambda \in \{U, N, D\}$.*

Proof. The upper bound follows from Theorem 10. The lower bound follows from Lemma 2, in the same way as in the proof of Theorem 12. □

5 Discussion

We introduced and studied restricted flow games – an extension of the classical max-flow problem to graphs that are both labeled and game graphs. We studied different variants of these games. Each variant is parameterized by two parameters: (1) the outcome of the game, namely, whether it is the maximal or minimal decomposition, and (2) the branching mode of the specification automaton, namely, whether it is a UFA, NFA, or DFA. For each variant, we considered

four decision problems: calculating the outcome of given strategies, calculating the value of the game, and deciding the positivity of the two measures. Our results are summarized in Figure 5. Below we discuss two directions for future research: memoryful strategies, and no-tolerance to flow loss and waste.

Outcome	$\lambda = U$	$\lambda \in \{N, D\}$
MAX	NP-complete	
MAX^+		NLOGSPACE-complete
MIN	co-NP-complete	
MIN^+		

Value	$\lambda = U$	$\lambda \in \{N, D\}$
MAX	Σ_3^P-complete	
MAX^+		
MIN	Σ_2^P-complete	
MIN^+		

Fig. 5. The complexity of calculating the outcome and the value.

5.1 Memoryful strategies

A major challenge in our research arises from the ambiguity in determining the outcome, caused by the strategies being memoryless. We addressed this by considering two possible outcomes: maximal and minimal. Further research could examine a variant in which strategies do depend in the history of the flow. Below we discuss such a variant. Consider a setting in which the specification is given by a DFA, the history of a unit of flow can be abstracted by a single state in the DFA, namely, the state reached by the run of the DFA on the word corresponding to the history of the unit. Indeed, all histories that reach the same state are right-congruent. Then, a policy for a vertex v gets not just the number of units that enter v, but the history of each of the units. Note that a representation of such a strategy requires exponential space, which makes the approach infeasible. We can, however, consider policies for vertices in which the players do not specify how they direct the flow for every possible input. Instead, the policies give "symbolic instructions" on how to direct the flow. For example, a policy can prioritize the states in the DFA, that is, determine which unit of flow should be handled first, based on its history. Then, for each state in the DFA, the policy prioritizes the outgoing edges. Thus, given the incoming flow as a multiset of states in the DFA, the policy handles the most important units first, and tries to direct them to the highest priority edge that is not saturated. While this limits the control the players have in their vertices, the outcome of a game with such policies is determined, and can be calculated in polynomial time.

Alternatively, we could consider different settings, where the game is played *sequentially*, rather than in parallel. Thus, the game is played in rounds, and in each round, a single unit of flow traverses the network until it reaches a sink or the target vertex. The players have a different policy for each vertex

for each round, which maps the history of the current unit (given by a state in a DFA, as above) to an outgoing edge. In these settings, we can find the path traversed by each unit of flow, round-by-round, and check if it corresponds to a word that satisfies the specification, in polynomial time. Thus, we do not need the maximal and minimal variants, as the paths traversed by the units of flow are determined. Finally, one could consider a probabilistic setting, where policies specify a distribution on the outgoing edges, and the goal of Player 0 is to maximize the expected value of the outcome.

5.2 No tolerance to loss and waste

Recall that "flow conservation", which holds in all vertices in the traditional definition of flow, does not hold in the game setting. Indeed, flow gets lost in vertices (in particular, sinks) whose incoming flow is greater than the total capacity of their outgoing edges. In restricted flow games, there is an additional notion of loss, which we term *waste*: A unit of flow is wasted if it reaches the target vertex, but does so along a route that does not satisfy the specification. In some cases, loss or waste cannot be tolerated, and the authority prefers to reduce the flow in order to avoid them. For example, when leaks are hazardous or reveal sensitive information, or when flow that reaches the target while violating the specification contaminates the rest of the flow or conflicts with the privacy requirements of the network.

In order to cope with such settings, it is interesting to study a *no-loss* and *no-waste* variants of restricted flow games, where the authority is limited to strategies that ensure no loss and/or no waste of flow.

References

1. S. Abiteboul and V. Vianu. Regular path queries with constraints. *J. Comput. Syst. Sci.*, 58(3):428–452, 1999.
2. R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: Theory, algorithms, and applications*. Prentice Hall Englewood Cliffs, 1993.
3. R. Alon and O. Kupferman. Mutually accepting capacitated automata. In *Proc. 22nd International Workshop on Descriptional Complexity of Formal Systems*, Lecture Notes in Computer Science. Springer, 2020.
4. C. Barrett, R. Jacob, and M. Marathe. Formal-language-constrained path problems. *SIAM Journal on Computing*, 30(3):809–837, 2000.
5. V. Blue, J. Adler, and G. List. Real-time multiple-objective path search for in-vehicle route guidance systems. *Journal of the Transportation Research Board*, 1588:10–17, 1997.
6. D. Calvanese, G. de Giacomo, M. Lenzerini, and M.Y. Vardi. Reasoning on regular path queries. *SIGMOD Rec.*, 32(4):83–92, 2003.
7. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.
8. E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

9. S.A. Cook. Path systems and language recognition. In *Proc. 2nd ACM Symp. on Theory of Computing*, pages 70–72, 1970.
10. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.
11. E.A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Doll*, 11(5):1277–1280, 1970. English translation by RF. Rinehart.
12. S. Even. *Graph Algorithms, 2nd edition*. Cambridge University Press, 2011.
13. L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.
14. L.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton Univ. Press, Princeton, 1962.
15. Z. Füredi, D. Reimer, and A. Seress. Triangle-free game and extremal graph problems. *Congressus Numerantium*, 82:123–128, 1991.
16. M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. Freeman and Co., 1979.
17. A.V. Goldberg, É. Tardos, and R.E. Tarjan. Network flow algorithms. Technical report, DTIC Document, 1989.
18. A.V. Goldberg and R.E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, 1988.
19. S. Guha, O. Kupferman, and G. Vardi. Multi-player flow games. In *Proc. 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 104–112, 2018.
20. D. Hefetz, M. Krivelevich, A. Naor, and M. Stojaković. On saturation games. *European Journal of Combinatorics*, 41:315–335, 2016.
21. D. Hefetz, O. Kupferman, A. Lellouche, and G. Vardi. Spanning-tree games. In *43rd Int. Symp. on Mathematical Foundations of Computer Science*, volume 117 of *LIPIcs*, pages 35:1–35:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
22. V. Kann. Maximum bounded 3-dimensional matching is max-snp-complete. *Information Processing Letters*, 37(1):27–35, 1991.
23. O. Kupferman. Examining classical graph-theory problems from the viewpoint of formal-verification methods. In *Proc. 49th ACM Symp. on Theory of Computing*, page 6, 2017.
24. O. Kupferman and T. Tamir. Properties and utilization of capacitated automata. In *Proc. 34th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 29 of *LIPIcs*, pages 33–44. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2014.
25. O. Kupferman and G. Vardi. Eulerian paths with regular constraints. In *41st Int. Symp. on Mathematical Foundations of Computer Science*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, page 62:1, 2016.
26. O. Kupferman and G. Vardi. The unfortunate-flow problem. In *Proc. 45th Int. Colloq. on Automata, Languages, and Programming*, volume 107 of *LIPIcs*, pages 157:1–157:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
27. O. Kupferman, G. Vardi, and M.Y. Vardi. Flow games. In *Proc. 37th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 93 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:38–38:16, 2017.
28. A.O. Mendelzon and P.T. Wood. Finding regular simple paths in graph databases. *SIAM Journal on Computing*, 24(6):1235–1258, 1995.

29. C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. 2nd edition.
30. E. Petrank. The hardness of approximation: gap location. In *The 2nd Israel Symposium on Theory and Computing Systems*, pages 275–284, 1993.
31. L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.
32. R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.
33. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.