Computing stable outcomes in symmetric additively-separable hedonic games⋆

Martin Gairing and Rahul Savani

Department of Computer Science, University of Liverpool.
{gairing,rahul.savani}@liverpool.ac.uk

Abstract. We study the computational complexity of finding stable outcomes in hedonic games, which are a class of coalition formation games. We restrict our attention to symmetric additively-separable hedonic games, which are a nontrivial subclass of such games that are guaranteed to possess stable outcomes. These games are specified by an undirected edge-weighted graph: nodes are players, an outcome of the game is a partition of the nodes into coalitions, and the utility of a node is the sum of incident edge weights in the same coalition. We consider several stability requirements defined in the literature. These are based on restricting feasible player deviations, for example, by giving existing coalition members veto power. We extend these restrictions by considering more general forms of preference aggregation for coalition members. In particular, we consider voting schemes to decide if coalition members will allow a player to enter or leave their coalition. For all of the stability requirements we consider, the existence of a stable outcome is guaranteed by a potential function argument, and local improvements will converge to a stable outcome. We provide an almost complete characterization of these games in terms of the tractability of computing such stable outcomes. Our findings comprise positive results in the form of polynomial-time algorithms, and negative (`PLS`-completeness) results. The negative results extend to more general hedonic games.

1 Introduction

Hedonic games were introduced in the economics literature as a flexible model of coalition formation [13]. In a hedonic game, each player has preferences over coalitions and an outcome of the game is a partition of the players into coalitions. The defining feature of a hedonic game is that for a given outcome each player cares only about the other players in the same coalition. It is natural to judge the quality of an outcome by how stable it is with respect to the players' preferences. Many different notions of stability appear in the literature. The survey by Aziz and Savani [2] gives detailed background on hedonic games and outlines their applications. This paper studies the computational complexity of finding stable outcomes in hedonic games.

In this paper, we consider and extend the stability requirements for hedonic games introduced in the seminal work of Bogomolnaia and Jackson [7]. An outcome of a hedonic game is called *Nash-stable* if no player prefers to be in a different coalition. For Nash stability, the feasibility of a deviation depends only on the preferences of the deviating player. Less stringent stability requirements restrict feasible deviations: a coalition may try to hold on to an attractive player or block the entry of an unattractive player. In [7], deviations are restricted by allowing members of a coalition to

---

"veto" the entry or exit of a player. They introduce *individual stability*, where every member of a coalition has a veto that can prevent a player from joining (deviating to) this coalition, i.e., a player can deviate to another new coalition only if *everyone* in this new coalition is happy to have her. They also introduce *contractual individual stability*, where, in addition to a veto for entering, coalition members have a veto to prevent a player from leaving the coalition - a player can deviate only if everyone in her coalition is happy for her to leave.

The case where every member of a coalition has a veto on allowing players to enter and/or leave the coalition can be seen as an extreme form of *voting*. This motivates the study of more general voting mechanisms for allowing players to enter and leave coalitions. In this paper, we consider general voting schemes, for example, where a player is allowed to join a coalition if the majority of existing members would like the player to join. We also consider other methods of *preference aggregation* for coalition members. For example, a player is allowed to join a coalition only if the aggregate utility (i.e., the sum of utilities) existing members have for the entrant is non-negative. These preference aggregation methods are also considered in the context of preventing a player from leaving a coalition. We study the computational complexity of finding stable outcomes under stability requirements with various restrictions on deviations.

1.1 The model

In this paper, we study hedonic games with *symmetric additively-separable* utilities, which allow a succinct representation of the game as an *undirected edge-weighted graph* $G = (V, E, w)$. For clarity of our voting definitions, we assume w.l.o.g. that $w_e \neq 0$ for all $e \in E$ (an edge with weight 0 can be dropped). Every node $i \in V$ represents a player. An outcome is a partition p of V into coalitions. Denote by $p(i)$ the coalition to which $i \in V$ belongs under p, and by $E(p(i))$ the set of edges $\{\{i, j\} \in E \mid j \in p(i)\}$.

The utility of $i \in V$ under p is the sum of the weights of edges to others in the same coalition, i.e.,

$$\sum_{e \in E(p(i))} w(e).$$

Each player wants to maximize her utility, so a player *wants to deviate* if there exists a (possibly empty) coalition c where

$$\sum_{e \in E(p(i))} w(e) < \sum_{\{\{i,j\} \in E \mid j \in c\}} w(\{i, j\}).$$

We consider different restrictions on player deviations. They restrict when players are allowed to join and/or leave coalitions. A deviation of player i to coalition c is called

- *Nash feasible* if player i wants to deviate to c.
- *vote-in feasible* with threshold T_{in} if it is Nash feasible and either at least a T_{in} fraction of i's edges to c are positive or i has no edge to c.

- *vote-out feasible* with threshold T_{out} if it is Nash-feasible and either at least a T_{out} fraction of i's edges to $p(i)$ are negative or i has no edges within $p(i)$.
- *sum-in feasible* if it is Nash feasible and

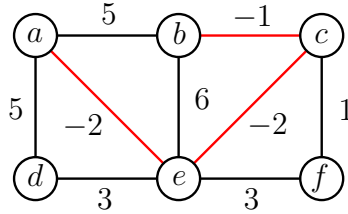$$\sum_{\{\{i,j\}\in E \,\mid\, j\in c\}} w(\{i,j\}) \geq 0.$$

- *sum-out feasible* if it is Nash feasible and

$$\sum_{e\in E(p(i))} w(e) \leq 0.$$

Outcomes where no corresponding feasible deviation is possible are called *Nash stable, vote-in stable, vote-out stable, sum-in stable*, and *sum-out stable*, respectively. Outcomes which are vote-in (resp. vote-out) stable with $T_{in} = 1$ (resp. $T_{out} = 1$) are also called *veto-in* (resp. *veto-out*) *stable*. Note that an outcome is veto-in stable iff it is an *individual stable* outcome; and an outcome is both veto-in and veto-out stable iff it is a *contractual individual stable* outcome (we use the terms *individual stable* and *contractual individual stable* since they are commonly used and known in the economics literature following their definition in [7]).

1.2 An example

Figure 1 gives an example of an additively-separable symmetric hedonic game that we use to illustrate some of the stability requirements that we have defined. Consider the outcome $\{\{a, b, d\}, \{c, e, f\}\}$. The utilities of the players a, b, c, d, e, f are $10, 5, -1, 5, 1, 4$, respectively.

Fig. 1. An example of an additively-separable symmetric hedonic game.

Players a, b, d, f have no Nash-feasible deviations, c has a Nash-feasible deviation to go alone and start a singleton coalition, and e has a Nash-feasible deviation to join the other coalition. The deviation of c is not veto-out feasible, since f prefers c to stay, however it is vote-out feasible for any $T_{out} \leq 0.5$. It is also sum-out feasible. The deviation of e is not veto-in feasible, but is vote-in feasible for any $T_{in} \leq 2/3$. Since there are no deviations that are both veto-in and veto-out feasible, this is a contractual individual stable outcome. The outcome $\{\{a, b, d\}, \{c\}, \{e, f\}\}$ is an individual stable outcome, and $\{\{a, b, d, e, f\}, \{c\}\}$ is Nash stable.

1.3 Justification of the model

With the goal of understanding how difficult it is for agents to *find* stable outcomes, we focus on a model in which they are *guaranteed* to exist. The computational complexity of a problem is measured in terms of the size of its input and therefore depends on the representation of the problem instance. For games, we desire that the size of the input is polynomial in the number of players, as this is the natural parameter with which to measure the size of the game. We consider only such *succinct representations*, since otherwise we can find solutions using trivial algorithms (enumeration of strategy profiles) that are polynomial in the input size. Our focus on additively-separable games is motivated by the hardness of even deciding the existence of stable outcomes and other solution concepts for more general (universal) succinct representations, such as hedonic nets [15]. A *non-symmetric* additively-separable game, which is represented by a edge-weighted *directed* graph, may not have a Nash-stable outcome [7, 5], and deciding existence is NP-complete. We study a more restrictive model where stable outcomes (for all of the stability requirements we consider) are guaranteed to exist, noting that our hardness results extend to all more general models where existence of stable outcomes is either guaranteed or promised, i.e., instances are restricted to those possessing stable outcomes.

In a *symmetric* additively-separable hedonic game, for each of the stability requirements we consider, a stable outcome *always exists* by a simple potential function argument: the potential function is the total happiness of an outcome, i.e., the sum of players' utilities. Unilateral player deviations improve the potential. So for all our considered stability requirements, local improvements will find a stable outcome, and all the problems we consider are in the complexity class PLS (polynomial local search) [23], which we introduce next.

1.4 Local search and the complexity class PLS

Local search is one of few general and successful approaches to difficult combinatorial optimisation problems. A local search algorithm tries to find an improved solution in the *neighborhood* of the current solution. A solution is *locally optimal* if there is no better solution in its neighborhood. Johnson et al. [23] introduced the complexity class PLS (polynomial local search) to capture those local search problems for which a better neighboring solution can be found in polynomial time if one exists, and a local optimum can be verified in polynomial time.

A problem in PLS comprises a finite set of candidate solutions. Every candidate solution has an associated non-negative integer cost, and a neighbourhood of candidate solutions. In addition, a PLS problem is specified by the following three polynomial-time algorithms that:

1. construct an initial candidate solution;
2. compute the cost of any candidate solution in polynomial time;

3. given a candidate solution, provide a neighbouring solution with lower cost if one exists.

The goal in a `PLS` problem is to find a local optimum, that is, a candidate solution whose cost is no more than the cost of any of its neighbours.

Suppose A and B are problems in `PLS`. Then A is `PLS`-reducible to B if there exist polynomial-time computable functions f and g such that f maps an instance I of A to an instance $f(I)$ of B, and g maps the local optima of instance $f(I)$ of B to local optima of instance I. A problem in `PLS` is `PLS`-complete if all problems in `PLS` are `PLS`-reducible to it. Prominent `PLS`-complete problems include finding a locally optimal max-cut in a graph (LOCALMAXCUT) [30], or a stable solution in a Hopfield network [23]. `PLS` captures the problem of finding pure Nash equilibria for many classes of games where pure Nash equilibria are guaranteed to exist, such as congestion games [18], for which is also `PLS`-complete to find a pure equilibrium.

On the one hand, finding a locally optimal solution is presumably easier than finding a global optimum; in fact, it is very unlikely that a `PLS` problem is `NP`-hard since this would imply `NP` = `coNP` [23]. On the other hand, a polynomial-time algorithm for a `PLS`-complete problem would resolve a number of long open problems, e.g., since it would show that *simple stochastic games* can be solved in polynomial time [34]. Thus, `PLS`-complete problems are believed not to admit polynomial-time algorithms.

1.5 Computational problems

We define the search problems, NASHSTABLE, IS (individual stable), CIS (contractual individual stable), VOTEIN, and VOTEOUT of finding a stable outcome for the respective stability requirement. We introduce VOTEINOUT as the search problem of finding an outcome which is vote-in and vote-out stable. All voting problems are parametrized by T_{in} and/or T_{out}. We recall that outcomes which are vote-in (resp. vote-out) stable with $T_{in} = 1$ (resp. $T_{out} = 1$) are also called *veto-in* (resp. *veto-out*) *stable*, so IS is the computational problem of finding a veto-in stable outcome, and CIS is the problem of finding an outcome that is both veto-in and veto-out stable. We also introduce SUMCIS as the problem of finding an outcome which is sum-in and sum-out stable.

Symmetric additively-separable hedonic games are closely related to party affiliation games, which are also specified by an undirected edge-weighted graph. In a party affiliation game each player must choose between one of two "parties"; a player's happiness is the sum of her edges to nodes in the same party; in a stable outcome no player would prefer to be in the other party. The problem PARTYAFFILIATION is to find a stable outcome in such a game. If such an instance has only negative edges then it is equivalent to the problem LOCALMAXCUT, which is to find a stable outcome of a local max-cut game. In party affiliation games there are at most two coalitions, while in hedonic games any number of coalitions is allowed. Thus, whereas PARTYAF-FILIATION for instances with only negative edges is `PLS`-complete [30], NASHSTABLE

is trivial in this case, as the outcome where all players are in singleton coalitions is Nash-stable. Both problems are trivial when all edges are non-negative, in which case the grand coalition of all players is Nash-stable. Thus, interesting hedonic games contain both positive and negative edges.

The problem OneEnemyPartyAffiliation is to find a stable outcome of a party affiliation game where each node is incident to at most one negative edge. In this paper, we use a variant of this problem as a starting point for some of our reductions:

Definition 1. *We define the problem* OneEnemyPartyAffiliation* *as a restricted version of* OneEnemyPartyAffiliation *which is restricted to instances where no player is ever indifferent between the two coalitions.*

We show in Corollary 1 (page 18) that OneEnemyPartyAffiliation* is PLS-complete.

1.6   Our results

In this paper, we examine the complexity of computing stable outcomes in symmetric additively-separable hedonic games. We observe that NashStable, i.e., the problem of computing a Nash-stable outcome, is PLS-complete (Observation 1). Here, we give a simple reduction from PartyAffiliation, which was shown to be PLS-complete in [30]. Our reduction relies on a method to ensure that all stable outcomes use exactly two coalitions (where in general there can be as many coalitions as players).

Moreover, we study IS, i.e., the problem of finding an individually-stable outcome. We show that if the outcome is restricted to contain at most two coalitions, an individually-stable outcome can be found in polynomial time (Proposition 1). This suggests that a reduction showing PLS-hardness for IS cannot be as simple as for NashStable: one would need to construct hedonic games that allow three or more coalitions.

In order to prove that IS is PLS-complete, we first define a restricted version of PartyAffiliation, called OneEnemyPartyAffiliation, in which each player dislikes at most one other player. Our main technical result is that OneEnemyPartyAffiliation is PLS-complete (Theorem 2). This reduction is from Circuit-Flip and is rather involved. The instances of OneEnemyPartyAffiliation that we produce via this reduction have the property that no player is ever indifferent between two coalitions. We then show that such instances can be reduced to IS.

Perhaps surprisingly, given the apparently restrictive nature of the stability requirement, we show that sumCIS is PLS-complete (Theorem 4).

In contrast, we show that the problem CIS of finding a contractually-individually-stable outcome can be solved in polynomial time. We make explicit two conditions in Propositions 2 and 3, both met in the case of CIS, that (individually) guarantee that local improvements converge in polynomial time. We use these propositions to

give further positive results for other combinations of restrictions, where either the entering or leaving restriction is veto based.

Enter / Leave	1: no restr.	2: sum-in	3: veto-in	4: vote-in
A: no restr.	NASHSTABLE PLS-complete Observation 1	PLS-complete Observation 1	IS PLS-complete Theorem 3	VOTEIN PLS-complete Theorem 5
B: sum-out	PLS-complete Theorem 4	SUMCIS PLS-complete Theorem 4	P Proposition 2	?
C: veto-out	P Proposition 3	P Proposition 3	CIS P Proposition 2 or 3	P Proposition 3
D: vote-out	VOTEOUT ? (see Theorem 6)	? (see Theorem 6)	P Proposition 2	VOTEINOUT P $(T_{in}, T_{out} > 0.5)$ Theorem 7

Fig. 2. Table showing the computational complexity of the search problems for different entering and leaving deviation restrictions. Note that columns 1 and 2 are essentially equivalent, since if a player has a Nash-feasible deviation that results in a negative payoff, she also has a sum-in feasible (and hence also Nash-feasible) deviation, namely to form a singleton coalition.

Finally, we study the complexity of finding vote-in and vote-out stable outcomes. Using a different argument to the polynomial-time cases mentioned previously, we show that local improvements converge in polynomial time in the case of vote-in- and vote-out- stability with $T_{in}, T_{out} \geq 0.5$ and $T_{in} + T_{out} > 1$ (Theorem 7). We show that if we require vote-in-stability alone, we get a PLS-complete search problem (Theorem 5). The problem of finding a vote-out stable outcome is conceptually different (e.g., we can find a veto-out-stable outcome in polynomial time, whereas it is PLS-complete to find a veto-in-stable outcome). The technical difficulty in proving a hardness result for VOTEOUT is restricting the number of coalitions. Ultimately, we leave the complexity of VOTEOUT open, but do show that k-VOTEOUT, which is the problem of computing a vote-out stable outcome when at most k coalitions are allowed, is PLS-complete (Theorem 6).

Our results are summarized in Figure 2, which gives an almost complete characterization of tractability.

1.7 Related work

Hedonic coalition formation games were first considered by Dreze and Greenberg [13]. Greenberg [21] later surveyed coalition structures in game theory and economics. Based on [13], Bogomolnaia and Jackson [7] formulated different stability concepts in

the context of hedonic games - see also the survey [32]. These stability concepts were our motivation to introduce definitions of stability based on voting and aggregation.

The general focus in the game theory community has been on characterizing the conditions for which stable outcomes exist. Burani and Zwicker [9] showed that additively-separable and symmetric preferences guarantee the existence of a Nash-stable outcome. They also showed that under certain different conditions on the preferences, the set of Nash-stable outcomes can be empty but the set of individually-stable partitions is always non-empty.

Ballester [5] showed that for hedonic games represented by an *individually rational list of coalitions*, the complexity of checking whether core-stable, Nash-stable or individual-stable outcomes exist is NP-complete, and that every hedonic game has a contractually-individually-stable solution. Sung and Dimitrov [33] showed that for additively-separable hedonic games checking whether a core-stable, strict-core-stable, Nash-stable or individually-stable outcome exists is NP-hard. For core-stable and strict-core-stable outcomes those NP-hardness results have been extended by Aziz et al. [3] to the case of symmetric player preferences. Recently, the paper [28] unifies and extends several of these results by identifying simple conditions on expressivity of hedonic games that are sufficient for the problem of checking whether a given game admits a stable outcome to be computationally hard.

Branzei and Larson [8] studied the tradeoff between stability and social welfare in additively-separable hedonic games. Elkind and Wooldridge [15] characterize the complexity of problems related to coalitional stability for hedonic games represented by hedonic nets, a succinct, rule-based representation based on marginal contribution nets (introduced by Ieong and Shoham [22]). Cechlárová [10] surveys algorithmic problems related to stable outcomes.

The definition of party affiliation games we use appears in Balcan et al. [4]. Recent work on local max cut and party affiliation games has focused on approximation [6, 11]; see also [27]. For surveys on the computational complexity of local search, see [26, 1]. Our PLS-hardness results use ideas from Krentel [24], Schäffer and Yannakakis [30], Monien and Tscheuschner [25], and in particular Elsässer and Tscheuschner [17]. We use the PLS-completeness of LOCALMAXCUT which was shown in Schäffer and Yannakakis [30].

There is an extensive literature on weighted voting games, which are formally simple coalitional games. For such a game, a "solution" is typically a vector (or set of vectors) of payoffs for the players, rather than a coalition structure as in our setting; for recent work on computational problems associated with weighted voting games see [14, 16]. Deng and Papadimitriou [12] examined the computational complexity of computing solutions for coalitional games for a model similar to additively-separable hedonic games, where the game is given by an edge-weighted graph, and the value of a coalition of nodes is the sum of weights of edges in the corresponding subgraph.

1.8 Outline of the paper

In Section 2, we show that NASHSTABLE is PLS-complete. In Section 3, we prove our main technical result: ONEENEMYPARTYAFFILIATION is PLS-complete. ONEENEMY-PARTYAFFILIATION is the starting point for our reduction to IS (i.e., VETOIN) in Section 4, which shows that it too is PLS-complete. In Section 5, we show that the remaining veto-based problems, namely all those (except for IS) in row C and column 3 in Figure 2, can be solved in polynomial time. In Section 6, we show that the problem SUMCIS is PLS-complete. In Section 7, we give both positive and negative results for computing stable outcomes under various voting-based stability requirements. Finally, in Section 8, we conclude with open problems.

2 Nash stability and restricting the number of coalitions

In this section, we show that NASHSTABLE is PLS-complete via a reduction from PARTYAFFILIATION. Recall that in Nash-stable outcomes for hedonic additive separable games, players might form more than two coalitions while party affiliation games are restricted to two coalitions. To deal with this we use a mechanism, called *supernodes*, which can restrict the number of coalitions that will be non-empty in stable outcomes. In the reduction in this section we use two supernodes to restrict to two coalitions; later in the paper we will use a variable number of k supernodes to restrict the number of coalitions to k.

Observation 1 NASHSTABLE *is PLS-complete.*

Proof. Consider an instance of PARTYAFFILIATION, represented as an edge-weighted graph $G = (V, E, w)$. We augment G by introducing two new players, called *supernodes*. Every player $i \in V$ has an edge of weight $W > \sum_{e \in E} |w_e|$ to each of the supernodes. The two supernodes are connected by an edge of weight $-M$, where $M > |V| \cdot W$. By the choice of M the two supernodes will be in different coalitions in any Nash-stable outcome of the resulting hedonic game. Moreover, by the choice of W, each player will be in a coalition with one of the supernodes. So, in every Nash-stable outcome we have exactly two coalitions. The fact that edges to supernodes have all the same weight directly implies a one-to-one correspondence between the Nash-stable outcomes in the hedonic game and in the party affiliation game. $\square$

3 Key technical result: ONEENEMYPARTYAFFILIATION is PLS-complete

In this section, we prove our key technical result, that ONEENEMYPARTYAFFILIATION is PLS-complete. The instances that we produce have the useful property that no player is every indifferent between two coalitions, which we make explicit in Corollary 1. We use these special instances for other reductions in this paper.

The starting point for our reduction to ONEENEMYPARTYAFFILIATION is the prototypical PLS-complete problem CIRCUITFLIP, introduced and shown to be PLS-complete in [23].

Definition 2. *An instance of* CIRCUITFLIP *is a boolean circuit with n inputs and n outputs. A feasible solution is an assignment to the inputs and the value of a solution is the output treated as a binary number. The neighbourhood of an assignment consists of all assignments obtained by flipping exactly one input bit. The objective is to maximise the value.*

Theorem 2. ONEENEMYPARTYAFFILIATION *is PLS-complete.*

Proof. We reduce from CIRCUITFLIP. Let C be an instance of CIRCUITFLIP with inputs $V_1, \ldots, V_n$, outputs $C_1, \ldots, C_n$, and gates $G_1, \ldots, G_N$. We make the following simplifying assumptions about C: (i) The gates are topologically ordered so that if the output of G_i is an input to G_j then $i > j$. (ii) All gates are NOR gates with fan-in 2. (iii) $G_1, \ldots, G_n$ is the output and $G_{n+1}, \ldots, G_{2n}$ is the (bitwise) negated output of C with G_1 and G_{n+1} being the most significant bits. (iv) $G_{2n+1}, \ldots, G_{3n}$ outputs a (canonical) better neighbouring solution if $V_1, \ldots, V_n$ is not locally optimal.

We use two complete copies of C. One of them represents the current solution while the other ones represents the next (better) solution. Each copy gives rise to a graph. We will start by describing our construction for one of the two copies and later show how they interact. Given C construct a graph G_C as follows:

We have nodes $v_1, \ldots, v_n$ representing the inputs of C, and nodes g_i representing the output of the gates of C. We will also use g_i to refer to the whole gate. For $i \in [n]$, denote by $w_i := g_{2n+i}$ the nodes representing the better neighbouring solution. Recall that $g_1, \ldots, g_n$ represent the output of C while $g_{n+1}, \ldots, g_{2n}$ correspond to the negated output.

In our party affiliation game we use 0 and 1 to denote the two coalitions. We slightly abuse notation by using $u = \kappa$ for $\kappa \in \{0, 1\}$ to denote that node u is in coalition κ. In the construction, we assume the existence of nodes with a fixed coalition. This can be achieved as in the proof of Observation 1 with the help of supermodels. We use 0 and 1 to refer to those constant nodes. In the graphical representation (cf. Figure 3), we represent those constants by square nodes.

We follow the exposition of Schäffer and Yannakakis [30] and Elsässer and Tscheuschner [17] and use types to introduce our construction. Nodes may be part of multiple types. In general types are ordered w.r.t. decreasing edge weights. So earlier types are more important. Different types will serve different purposes.

Type 1: Check Gates. For each gate g_i we have a three-part component as depicted in Figure 3(a). The inputs of g_i, denoted $I_1(g_i)$ and $I_2(g_i)$, are either inputs of the circuit or outputs of some gate with larger index. The main purpose of this component is to check if g_i is correct, i.e., $g_i = \neg(I_1(g_i) \wedge I_2(g_i))$, and to set $z_i = 1$ if g_i is incorrect. The α, β, γ, δ and λ nodes are local nodes for the gate. A gate can be in two operational

modes, called *gate push regimes*. Type 7 will determine in which of the following push regimes a gate is.

Definition 3 (Gate push regimes). *In the* RESET GATE *regime* $\alpha_{i,1}$, $\alpha_{i,2}$, $\gamma_{i,1}$ *and* $\gamma_{i,2}$ *get a bias towards* 1 *while* $\lambda_{i,1}, \lambda_{i,2}, \beta_{i,1}, \beta_{i,2}, \beta_{i,3}, \delta_{i,1}, \delta_{i,2}$ *and* $\gamma_{i,3}$ *get a bias towards* 0. *In the* FIX GATE *regime we have opposite biases.*

Type 2: Propagate Flags. In order to propagate incorrect values for the z variables we interconnect them as in Figure 3(b) by using the topological order on the gates. Observe that for any locally optimal solution $z_i = 1$ enforces $z_j = 1$ for all $j < i$. The component is also used to (help to) fix the gates in order and to RESET them in the opposite order. Node z_{N+1} is for technical convenience.

Type 1 and 2 components are the same for both copies. In the following we describe how the copies interact. We denote the two copies of C by C^0 and C^1 and also use superscripts to distinguish between them for nodes of type 1 and 2.

Type 3: Set/Reset Circuits. The component of type 3 interconnects the z-flags from the two circuits C^0, C^1. This component is depicted in Figure 3(c) and has multiple purposes. First, it ensures that in a local optimum d^0 and d^1 are not both 1. Second, at the appropriate time, it triggers to reset the circuit with smaller output. And third, it locks d^0 or d^1 to 1 and resets them back to 0 at the appropriate times.

The z and y nodes can also be in two different operational modes called COMPUTE regime and RESET regime which is determined by Type 6.
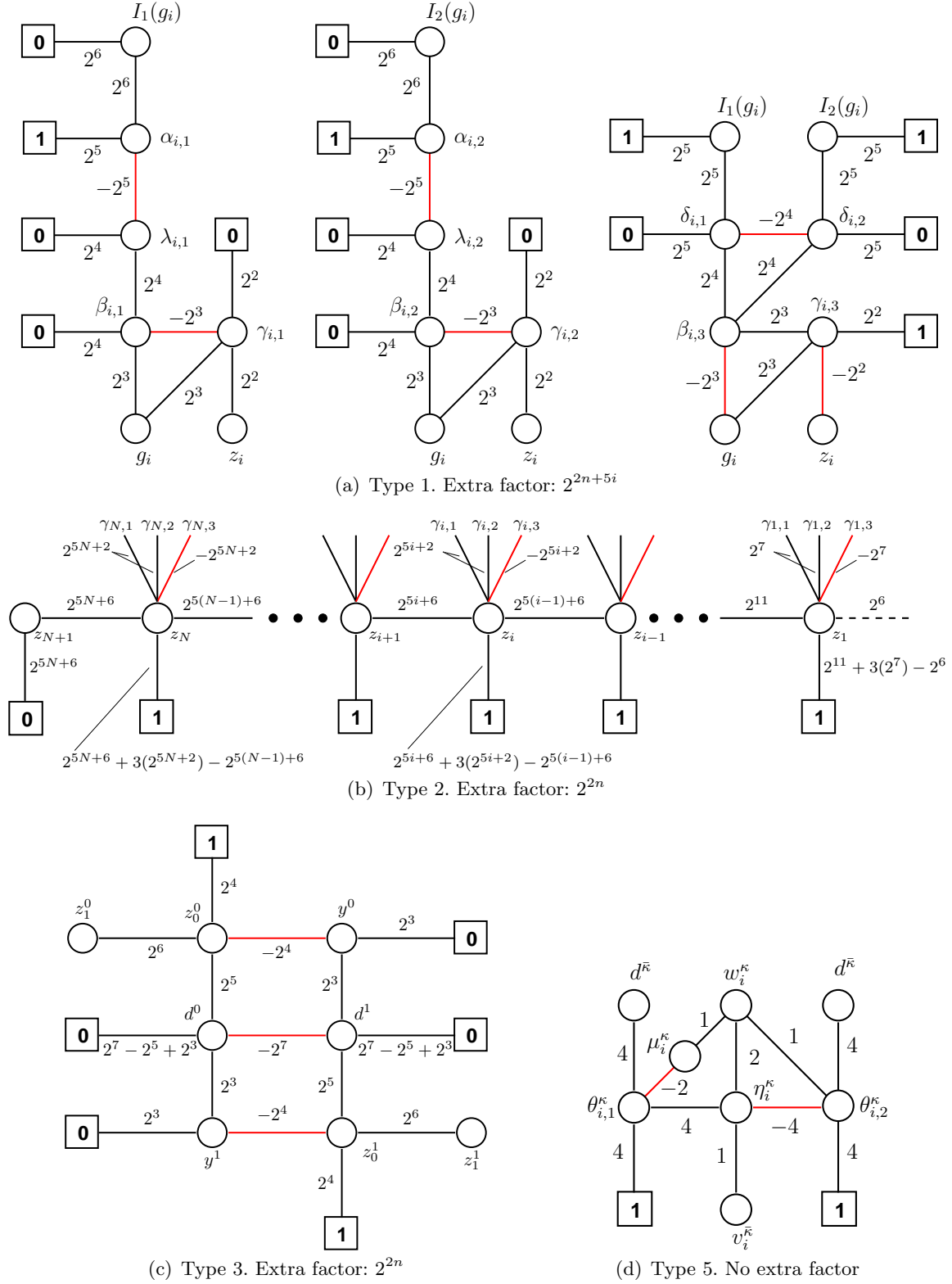
Definition 4 (Circuit push regimes). *Let* $\kappa \in \{0, 1\}$. *In the* COMPUTE *regime for* z^κ *all* z_i^κ *get a bias to* 0 *for all* $0 \le i \le N + 1$ *and* y^κ *gets a bias to* 1. *In the* RESET *regime for* z^κ *we give opposite biases.*

Type 4: Check Outputs. This component compares the current output of the two circuits and gives incentive to set one of the nodes d^0 or d^1 to 1 for which the output of the corresponding circuit is smaller. For all $i \in [n]$, we have edges $(d^0, g_{n+i}^0), (d^0, g_i^1), (d^1, g_{n+i}^1)$, (d^1, g_i^0) and $(0, g_{n+i}^0), (1, g_i^1), (0, g_{n+i}^1), (1, g_i^0)$ of weight 2^{2n+1-i}. To break symmetry we have edges $(0, d^0), (1, d^1)$ of weight 2^n.

Type 5: Feedback Better Solution. This component is depicted in Figure 3(d). It is used to feedback the improving solution of one circuit to the input of the other circuit. Its operation is explained in Lemma 2.

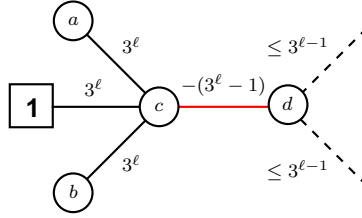For the remaining types we use the following lemma and definition which are analogous to those in [17, 25].

Lemma 1. *For any polynomial-time computable function* $f : \{0, 1\}^k \mapsto \{0, 1\}^m$ *one can construct a graph* $G_f(V_f, E_f, w)$ *having the following properties: (i) there exist* $s_1, \ldots, s_k, t_1, \ldots, t_m \in V_f$ *with no negative incident edge, (ii) each node in* V_f *is only incident to at most one negative edge, (iii)* $f(s) = t$ *in any Nash-stable solution of the party affiliation game defined by* G_f.

(a) Type 1. Extra factor: 2^{2n+5i}

(b) Type 2. Extra factor: 2^{2n}

(c) Type 3. Extra factor: 2^{2n}

(d) Type 5. No extra factor

Fig. 3. Components of type 1,2,3, and 5. Edge weights have to be multiplied by the factors given above.

Proof. It is well known that for any polynomial computable function $f : \{0,1\}^k \mapsto \{0,1\}^m$ one can construct a circuit C with polynomial many gates that implements this function [31, Theorem 9.30]. Clearly, we can also restrict C to NOR gates with fan-in and fan-out at most 2. Organize the gates in levels according to their distance to C's output; output gates are at level 1.

We replace each gate g_i at level ℓ with the gadget below. Nodes a, b are inputs and d is the output of the gate.

Fig. 4. NOR gate

If a (or b) is an input of the circuit then we connect a to the corresponding input s-node by an edge of weight $3^{\ell+1}$. If $\ell = 1$, i.e., g_i is an output gate, then we connect d to the corresponding output t-node with an edge of weight 1. Otherwise ($\ell > 1$), d is also the input to at most 2 lower level gates. The corresponding edges have weight at most $3^{\ell-1}$. In any Nash-stable solution, $d = 1$ if and only if $a = b = 0$. In other words $d = NOR(a, b)$. The claim follows since our construction fulfils properties (i), (ii) and (iii). □

Definition 5. *For a polynomial-time computable function $f : \{0,1\}^k \mapsto \{0,1\}^m$ we say that G_f as constructed in Lemma 1 is a graph that* looks at *$s_1, \ldots, s_k \in V_f$ and* biases *$t_1, \ldots, t_m \in V_f$ according to the function f.*

In the final three types we look at and bias nodes from the lower types already defined. For the final types we do not give explicit edge weights. In order that the "looking" has no side-effects on the operation of the lower types, we scale edge weights in these types such that any edge weight of lower type is larger than the sum of the edge weights of all higher types. More precisely, for $j \in \{5, 6, 7\}$, the weight of the smallest edge of type j is larger than the sum of weights of all edges of types $(j + 1), \ldots, 8$.

In the following, denote by $C(v)$ the value of circuit C of the CIRCUITFLIP instance on input $v = (v_i)_{i \in [n]}$ and $w(v)$ the better neighbouring solution. Both are functions as in Definition 5.

Type 6: Change Push Regimes for z. The component of type 6 looks at v^0, v^1, d^0, d^1, η^0 and η^1 (type 5) and biases z_i^0, z_i^1, y^0 and y^1 as follows. z^0 is put in the COMPUTE regime if at least one of the following 3 conditions is fulfilled: (i) $C(v^0) \geq C(v^1)$, (ii) $w(v^1) = v^0$, or (iii) $w(v^1) \neq \eta^1 \wedge d^0 = 1$. Else z^0 is put into the RESET regime.

Likewise z^1 is put in the COMPUTE regime if at least one of the following three conditions is fulfilled: (i) $C(v^0) < C(v^1)$, (ii) $w(v^0) = v^1$, or (iii) $w(v^0) \neq \eta^0 \wedge d^1 = 1$. Else z^1 is put into the RESET regime. Note that conditions (i) and (ii) are important for normal computation, while (iii) is needed to overcome bad starting configurations.

Type 7: Change Push Regimes for Gates. For each $i \in [N]$ and $\kappa \in \{0, 1\}$, if $z_{i+1}^\kappa = 0$ we put the local variable of g_i^κ in the FIX GATE regime and in the RESET GATE regime otherwise.

Type 8: Fix Incorrect Gate. For each $i \in [N]$ and $\kappa \in \{0, 1\}$, the components of type 8 give a tiny offset to g_i^κ for computing correctly. For each gate g_i^κ we look at $\alpha_{i,1}^\kappa, \alpha_{i,2}^\kappa$ and bias g_i^κ to $\neg(\alpha_{i,1}^\kappa \wedge \alpha_{i,2}^\kappa)$.

This completes our construction. We proceed by showing properties of Nash-stable outcomes. Each of the following six lemmas should be read with the implicit clause: "*In every Nash-stable outcome.*"

Lemma 2. *Let $\kappa \in \{0, 1\}$, then the following holds for all $i \in [n]$:*
(a) If $d^{\bar\kappa} = 0$ then w_i^κ is indifferent w.r.t. edges of type 5.
(b) If $d^{\bar\kappa} = 1$ then $\eta_i^\kappa = w_i^\kappa$.

Proof. Suppose $d^{\bar\kappa} = 0$. If $\eta_i^\kappa = 0$ then $\theta_{i,1}^\kappa = 1$ and $\theta_{i,2}^\kappa = \mu_i^\kappa = 0$, and hence w_i^κ is indifferent w.r.t. edges of type 5. The case $\eta_i^\kappa = 1$ is symmetric. This proves (a). If $d^{\bar\kappa} = 1$ then $\theta_{i,1}^\kappa = \theta_{i,2}^\kappa = 1$, so η_i^κ is indifferent w.r.t. the edges connecting it to $\theta_{i,1}^\kappa$ and $\theta_{i,2}^\kappa = 1$. Hence η_i^κ will copy the value of w_i^κ, which proves (b). $\square$

Lemma 3. *If g_i^κ is incorrect then $z_i^\kappa = 1$. If $z_i^\kappa = 1$ then $z_j^\kappa = 1$ for all $0 \leq j \leq i$ and $y^\kappa = 0$.*

Proof. Gate g_i^κ can be incorrect in two ways:

(i) $I_j(g_i^\kappa) = 1$ for some $j \in \{1, 2\}$ and $g_i^\kappa = 1$,
(ii) $I_1(g_i^\kappa) = I_2(g_i^\kappa) = 0$ and $g_i^\kappa = 0$.

For case (i) observe that $I_j(g_i^\kappa) = 1 \implies \alpha_{i,j}^\kappa = 1 \implies \lambda_{i,j} = 0 \implies \beta_{i,j}^\kappa = 0$. Together with $g_i^\kappa = 1$ this directly implies $\gamma_{i,j}^\kappa = 1$. Consider now case (ii). Since $I_1(g_i^\kappa) = I_2(g_i^\kappa) = 0$ we have $\delta_{i,1}^\kappa = \delta_{i,2}^\kappa = 0$ and therefore $\beta_{i,3}^\kappa = 0$. Together with $g_i^\kappa = 0$ this directly implies $\gamma_{i,3}^\kappa = 0$. In either case, this implies $z_i^\kappa = 1$, proving the first part of the lemma.

The second claim holds by induction, since $z_i^\kappa = 1$ enforces $z_{i-1}^\kappa = 1$, while $z_0^\kappa = 1$ enforces $y^\kappa = 0$. $\square$

Lemma 4. *If $z_{i+1}^\kappa = 1$ then the inputs $I_1(g_i^\kappa)$ and $I_2(g_i^\kappa)$ are indifferent with respect to the type 1 edges of gate g_i^κ.*

Proof. We show that $\alpha_{i,1}^\kappa = \alpha_{i,2}^\kappa = 1$ and $\delta_{i,1}^\kappa = \delta_{i,2}^\kappa = 0$, which implies the claim. According to type 7, since $z_{i+1}^\kappa = 1$, gate i is in the RESET GATE regime.

We first show that in this regime, we must have $\alpha_{i,1}^\kappa = \alpha_{i,2}^\kappa = 1$. It is immediate that if $I_j(g_i^\kappa) = 1$ then $\alpha_{i,j}^\kappa = 1$ for $j \in \{1,2\}$. We now show that $I_j(g_i^\kappa) = 0$ implies $\alpha_{i,j}^\kappa = 1$ for $j \in \{1,2\}$. Suppose, without loss of generality, that $j = 1$. Suppose $I_1(g_i^\kappa) = 0$ and for the sake of contradiction that $\alpha_{i,1}^\kappa = 0$. Since $\alpha_{i,1}^\kappa$ is biased to 1 it can only be 0 if $\lambda_{i,1}^\kappa = 1$. Since $\lambda_{i,1}^\kappa$ is biased to 0, it can only be 1 if $\beta_{i,1}^\kappa = 1$. Since $\beta_{i,1}^\kappa$ is biased to 0, it can only be 1 if $g_i = 1$ and $\gamma_{i,1}^\kappa = 0$. However, since $\beta_{i,1}^\kappa = 1$ and $g_i^\kappa = 1$ and $\gamma_{i,1}^\kappa$ is biased to 1, we have $\gamma_{i,1}^\kappa = 1$. Thus $\alpha_{i,1}^\kappa = 1$.

We are left to show that in the RESET GATE regime we must have $\delta_{i,1}^\kappa = \delta_{i,2}^\kappa = 0$. If $(I_1(g_i^\kappa), I_2(g_i^\kappa)) = (0,0)$, it is immediate that $\delta_{i,1}^\kappa = \delta_{i,2}^\kappa = 0$. Suppose $(I_1(g_i^\kappa), I_2(g_i^\kappa)) = (0,1)$. Then it is immediate that $\delta_{i,1}^\kappa = 0$, and then since $\delta_{i,2}^\kappa$ is biased to 0, it must also be 0. The case $(I_1(g_i^\kappa), I_2(g_i^\kappa)) = (1,0)$ is symmetric. Finally, suppose $(I_1(g_i^\kappa), I_2(g_i^\kappa)) = (1,1)$. If $\delta_{i,1}^\kappa = \delta_{i,2}^\kappa = 1$ then it is immediate that $\beta_{i,3}^\kappa = 1$. Then $\delta_{i,1}^\kappa$ and $\delta_{i,2}^\kappa$ are both indifferent to the edges of type 1, but they are not stable since they are biased to 0. Suppose $\delta_{i,1}^\kappa = 1$ and $\delta_{i,2}^\kappa = 0$. If $\beta_{i,3}^\kappa = 0$ then $\delta_{i,1}^\kappa$ is indifferent to the edges of type 1, but is biased to 0 and hence is not stable. If $\beta_{i,3}^\kappa = 1$, then since $\beta_{i,3}^\kappa$ is biased to 0, we must have $g_i = 0$ and $\gamma_{i,3}^\kappa = 1$. But then $\gamma_{i,3}^\kappa$ is not stable since it is biased to 0. The case $\delta_{i,1}^\kappa = 0$ and $\delta_{i,2}^\kappa = 1$ is symmetric. $\square$

Lemma 5. *Suppose $z_{i+1}^\kappa = 0$ and $z_i^\kappa = 1$ for some index $1 \le i \le N$.*
(a) If g_i^κ is correct then $\gamma_{i,1}^\kappa = \gamma_{i,2}^\kappa = 0$ and $\gamma_{i,3}^\kappa = 1$.
(b) If g_i^κ is not correct then g_i^κ is indifferent w.r.t. edges of type 1 but w.r.t. the edges only in type 8 deviating would improve her happiness.

Proof. According to type 7, since $z_{i+1}^\kappa = 0$, gate i is in the FIX GATE regime. Thus $\gamma_{i,1}^\kappa$ and $\gamma_{i,2}^\kappa$ are biased to 0. First suppose the gate is correct.

If the correct output is 0 then we have $g_i^\kappa = 0$, and $z_i^\kappa = 1$ by assumption. Then $\gamma_{i,1}^\kappa$ and $\gamma_{i,2}^\kappa$ either prefer 0 or are indifferent w.r.t. the edges in type 1 (depending on the values of $\beta_{i,1}^\kappa$ and $\beta_{i,2}^\kappa$). As they are biased to 0 they will be 0. Suppose that $\gamma_{i,3}^\kappa = 0$ for the sake of contradiction. Then since it is biased to 1, we must have $\beta_{i,3}^\kappa = 0$. Since $\beta_{i,3}^\kappa$ is biased to 1, we must have $\delta_{i,1}^\kappa = \delta_{i,2}^\kappa = 0$. However, as the correct output is 0, at least one of the input bits must be 1. Suppose w.l.o.g. that $I_1(g_i^\kappa) = 1$. Then, since $\delta_{i,1}^\kappa$ is indifferent w.r.t. the edges in type 1 and is biased to 1, it must be 1, a contradiction.

Now suppose the correct output is 1. Thus we have $g_i^\kappa = 1$, and $z_i^\kappa = 1$ by assumption. Then $\gamma_{i,3}^\kappa$ either prefers 1 or is indifferent w.r.t. the edges in type 1 (depending on the value of $\beta_{i,3}^\kappa$). Since $\gamma_{i,3}^\kappa$ is biased to 1, it will be 1. Suppose that $\gamma_{i,1}^\kappa = 1$ for the sake of contradiction. Since $\gamma_{i,1}^\kappa$ is biased to 0 it can only be 1 if $\beta_{i,1}^\kappa = 0$. Since $\beta_{i,1}^\kappa$ is biased to 1 is can only be 0 if $\lambda_{i,1}^\kappa = 0$. Since $\lambda_{i,1}^\kappa$ is biased to 1 it can only be 0 if $\alpha_{i,1}^\kappa = 1$. Since the output is 1 the input $I_1(g_i^\kappa) = 0$, and then since $\alpha_{i,1}^\kappa$ is indifferent w.r.t. the edges in type 1 and is biased to 0, it must be 0, a contradiction. The same reasoning applies for $\gamma_{i,2}^\kappa$. This completes the proof of (a).

Now suppose the output is incorrect. Note that g_i is indifferent w.r.t. the edges of type 1 if and only if $\beta_{i,1}^\kappa \neq \gamma_{i,1}^\kappa$ and $\beta_{i,2}^\kappa \neq \gamma_{i,2}^\kappa$ and $\beta_{i,3}^\kappa = \gamma_{i,3}^\kappa$.

First suppose the output is 0. Thus we have $g_i^\kappa = 0$, and $z_i^\kappa = 1$ by assumption. Since the output is 0 and incorrect, we have $I_1(g_i^\kappa) = I_2(g_i^\kappa) = 0$,

Suppose $\beta_{i,1}^\kappa = \gamma_{i,1}^\kappa = 1$. Then $\gamma_{i,1}^\kappa$ is indifferent w.r.t. the edges in type 1 and is biased to 0, a contradiction. Now suppose $\beta_{i,1}^\kappa = \gamma_{i,1}^\kappa = 0$. Since $\beta_{i,1}^\kappa$ is biased to 1, we have $\lambda_{i,1}^\kappa = 0$. Since $\lambda_{i,1}^\kappa$ is biased to 0, we have $\alpha_{i,1}^\kappa = 1$. But $\alpha_{i,1}^\kappa$ is indifferent w.r.t. the edges of type 1 and biased to 0, a contradiction. Thus $\beta_{i,1}^\kappa \neq \gamma_{i,1}^\kappa$ and likewise $\beta_{i,2}^\kappa \neq \gamma_{i,2}^\kappa$.

Now suppose $\beta_{i,3}^\kappa = 1$ and $\gamma_{i,3}^\kappa = 0$. Then $\gamma_{i,3}^\kappa$ is indifferent w.r.t. the edges in type 1 and biased to 1, a contradiction. Now suppose $\beta_{i,3}^\kappa = 0$ and $\gamma_{i,3}^\kappa = 1$. Then $\gamma_{i,3}^\kappa$ prefers to be 0 than 1, a contradiction. We have shown that g_i^κ is indifferent w.r.t. edges of type 1 when the output is 0 and incorrect.

Since $I_1(g_i^\kappa) = I_2(g_i^\kappa) = 0$, and $\alpha_{i,1}^\kappa$ and $\alpha_{i,2}^\kappa$ are biased to 0, we have $\alpha_{i,1}^\kappa = \alpha_{i,2}^\kappa = 0$. Thus type 8 biases g_i^κ to 1, and it would gain by flipping as claimed.

Now suppose the output is 1. Thus we have $g_i^\kappa = 1$, and $z_i^\kappa = 1$ by assumption. Suppose $\beta_{i,1}^\kappa = \gamma_{i,1}^\kappa = 0$. Then $\gamma_{i,1}^\kappa$ prefers to be 1 than 0, a contradiction. Suppose $\beta_{i,1}^\kappa = \gamma_{i,1}^\kappa = 1$. Then $\gamma_{i,1}^\kappa$ is indifferent w.r.t. edges of type 1 and is biased to 0, a contradiction. Thus $\beta_{i,1}^\kappa \neq \gamma_{i,1}^\kappa$ and likewise $\beta_{i,2}^\kappa \neq \gamma_{i,2}^\kappa$.

Now suppose $\beta_{i,3}^\kappa = 1$ and $\gamma_{i,3}^\kappa = 0$. Then $\gamma_{i,3}^\kappa$ prefers to be 1 than 0, a contradiction. Now suppose $\beta_{i,3}^\kappa = 0$ and $\gamma_{i,3}^\kappa = 1$. Since $\beta_{i,3}^\kappa$ is biased to 1, we must have $\delta_{i,1}^\kappa = \delta_{i,2}^\kappa = 0$. Since the output is 1 and incorrect, we have at least one of $I_1(g_i^\kappa)$ and $I_2(g_i^\kappa)$ equal to 1. Suppose w.l.o.g. that $I_1(g_i^\kappa) = 1$. Then, since $\delta_{i,1}^\kappa$ is indifferent w.r.t. the edges in type 1 and is biased to 1, it must be 1, a contradiction. We have shown that g_i^κ is indifferent w.r.t. edges of type 1 when the output is 1 and incorrect.

At least one of $I_1(g_i^\kappa)$ and $I_2(g_i^\kappa)$ are 1. Suppose w.l.o.g. that $I_1(g_i^\kappa) = 1$. Then $\alpha_{i,1}^\kappa = 1$. Thus type 8 biases g_i^κ to 0, and it would gain by flipping as claimed. This completes the proof of (b). □

Lemma 6. *If $d^\kappa = 1$ and $d^{\overline{\kappa}} = 0$ then for all $1 \leq i \leq 2n$, node g_i^κ is indifferent w.r.t. edges in type 4.*

Proof. Each of these nodes is incident to exactly two type 4 edges both having the same weight. For $1 \leq i \leq n$ these are $(1, g_i^\kappa)$ and $(d^{\overline{\kappa}}, g_i^\kappa)$, while for $n + 1 \leq i \leq 2n$ these are $(0, g_i^\kappa)$ and (d^κ, g_i^κ). The claim follows since $d^\kappa = 1$ and $d^{\overline{\kappa}} = 0$. □

Lemma 7. *Suppose $d^\kappa = 1$ and $d^{\overline{\kappa}} = 0$.*

(a) If z^κ is in the COMPUTE regime then $z_i^\kappa = 0$ for all $0 \leq i \leq N + 1$ and $y^\kappa = 1$.
(b) If z^κ is in the RESET regime then $z_i^\kappa = 1$ for all $0 \leq i \leq N + 1$ and $y^\kappa = 0$.

Proof. We start proving part (a). Since z^κ is in the COMPUTE regime we immediately get $z_{N+1}^\kappa = 0$. Assume, by way of contradiction, that there exists an index $1 \leq i \leq N$ such that $z_i^\kappa = 1$ and $z_{i+1}^\kappa = 0$. First assume g_i^κ is correct. Then

$\gamma_{i,1}^{\kappa} = \gamma_{i,2}^{\kappa} = 0$ and $\gamma_{i,3}^{\kappa} = 1$ by Lemma 5(a). This and the fact that z_{i+1}^{κ} is biased to 0 implies that $z_i^{\kappa} = 0$, a contradiction. Now assume g_i^{κ} is not correct. Then, by Lemma 5(b), g_i^{κ} is indifferent w.r.t. edges of type 1 but w.r.t. the edges only in type 8 flipping would improve her happiness. If $3n + 1 \le i \le N$ this already implies that g_i^{κ} would gain by switching, a contradiction. For the outputs and negated outputs, i.e., $1 \le i \le 2n$, we know by Lemma 6 that g_i^{κ} is indifferent w.r.t. edges in type 4. Moreover for the gates that represent the better neighbouring solution, i.e. $2n + 1 \le i \le 3n$, we know by Lemma 2(a) that g_i^{κ} is indifferent w.r.t. edges in type 5. (Recall that w_i^{κ} is just another name for g_{2n+i}^{κ}.) In either case, g_i^{κ} would gain by switching, a contradiction. Thus, $z_i^{\kappa} = 0$ for all $1 \le i \le N$. It remains to show that $z_0^{\kappa} = 0$ and $y^{\kappa} = 1$. Since $z_1^{\kappa} = 0$ and z^{κ} is in the COMPUTE regime, we get (by inspection of type 3 edges) that $z_0^{\kappa} = 0$ which then implies $y^{\kappa} = 1$. This completes the proof of part (a).

To see part (b), observe that $d^{\overline{\kappa}} = 0$ together with the bias of y^{κ} to 0 implies $y^{\kappa} = 0$. Now since $y^{\kappa} = 0$ and $d^{\kappa} = 1$ the bias of z_0^{κ} enforces $z_0^{\kappa} = 1$. The rest is by induction since $z_i^{\kappa} = 1$ and the bias directly implies $z_{i+1}^{\kappa} = 1$ for all $0 \le i \le N$. This completes the proof of part (b). $\qquad\square$

We now continue with the proof of Theorem 2. Suppose we are in a Nash-stable outcome of the party affiliation game. For our proof we assume $C(v^0) \ge C(v^1)$. We will point out the small differences of the other case afterwards. Since $C(v^0) \ge C(v^1)$, z^0 is in the COMPUTE regime, i.e., all z_i^0 are biased to 0 and y^0 is biased to 1 (by type 6). Thus, $z_{N+1}^0 = 0$.

The remainder of the proof splits depending on the coalition of z_1^0 and z_1^1. By Lemma 3 we know that $z_1^{\kappa} = 0$ implies that all gates in C^{κ} are correct.

$\underline{z_1^0 = 1}$: By Lemma 3 we have $z_0^0 = 1$ and $y^0 = 0$. If $d^0 = d^1 = 0$ then d^0 is better off changing to 1 (by inspection of type 3 edges). If $d^0 = 1$ then Lemma 7(a) implies $z_1^0 = 0$, a contradiction. If $d^1 = 1$ and z^1 is in the RESET regime then by Lemma 7(b) and Lemma 4, v^1 is indifferent w.r.t. type 1 edges. Thus $v^1 = \eta^0$. But then either condition (ii) or (iii) for putting z^1 in the COMPUTE regime (cf. type 6) are fulfilled. So z^1 has to be in the COMPUTE regime. Lemma 7(a) then implies $z_1^1 = 0$. But then the neighbourhood of d^1 in type 3 is dominated by 0, a contradiction to $d^1 = 1$.

$\underline{z_1^0 = 0 \text{ and } z_1^1 = 1}$: By Lemma 3 we have $z_0^1 = 1$ and $y^1 = 0$. Since $C(v^0) \ge C(v^1)$ we know that z^0 is in the COMPUTE regime. So $z_1^0 = 0$ enforces $z_0^0 = 0$ and $y^0 = 0$. By inspection of type 3 edges we have $d^0 = 0$ and thus $d^1 = 1$. First assume that z^1 is in the RESET regime, then $z_i^1 = 1$ for all $0 \le i \le N + 1$ and Lemma 4 says that the inputs of all gates g_i^1 are indifferent w.r.t. type 1 edges. In particular this holds for $v^1 = (v_i^1)_{i \in [n]}$, so $v^1 = \eta^0$. By Lemma 2(b), $\eta^0 = w^0$. Since $z_1^0 = 0$, C^0 is computing correctly and thus $w^0 = w(v^0)$. Combining this we get $v_1 = w(v^0)$ which contradicts our assumption that z^1 is in the RESET regime. Thus z^1 is in the COMPUTE regime. Since $d^1 = 1$ we can apply Lemma 7(a) to conclude $z_1^1 = 0$, a contradiction.

$z_1^0 = 0$ and $z_1^1 = 0$: By Lemma 3 we have $z_0^0 = z_0^1 = 0$ and $y^0 = y^1 = 1$. Moreover we know that both circuits are computing correctly. If $d^0 = 1$ then $d^1 = 0$ and d^0 is indifferent w.r.t. type 3 edges. Since both circuits are computing correctly and $C(v^0) \geq C(v^1)$, the type 4 edges enforce $d^0 = 0$. But then d^1 is indifferent w.r.t. type 3 edges and the type 4 edges enforce $d^1 = 1$. So, $d^0 = 0$ and $d^1 = 1$. If z^1 is in the RESET regime then Lemma 7(b) gives $z_1^1 = 1$, a contradiction. Thus, z^1 is in the COMPUTE regime. Since $d^1 = 1$ we can apply Lemma 2(b). This and the fact that C^0 is computing correctly implies $\eta^0 = w(v^0)$. So z^1 can only be in the COMPUTE regime if $v^1 = w(v^0)$. Since $C(v^0) \geq C(v^1)$ this implies that $v^0 = v^1$ is a local optimum for the circuit C.

This finishes the proof in case $C(v^0) \geq C(v^1)$. The case $C(v^0) < C(v^1)$ is completely symmetric except here the conclusion $v^0 = v^1$ in the very last sentence leads to the contradiction $C(v^0) < C(v^0)$. So this case can't happen in a local optimum.

Note that throughout the construction we made sure that no node is incident to more than one negative edge. This completes the proof of Theorem 2. $\qquad\square$

The instance produced by this reduction has the property that no node is indifferent between the two coalitions. We will make use of this property later in the paper.

Corollary 1. ONEENEMYPARTYAFFILIATION *is* PLS-*complete even if restricted to instances where no player is ever indifferent between the two coalitions, i.e.* ONE-ENEMYPARTYAFFILIATION* *is* PLS-*complete.*

4 Individual stability

In this section, we study the computational complexity of finding individual stable outcomes. We first provide a polynomial-time algorithm for 2-IS, which we define as the problem of finding an individual stable outcome when only two coalitions can form, i.e., we restrict the number of coalitions in the problem definition as for PARTYAFFILIATION. The main result in this section is that IS is PLS-complete. We reduce from ONEENEMYPARTYAFFILIATION that was shown to be PLS-complete in the previous section. Our reduction uses exactly 5 coalitions, a restriction which we enforce using supernodes. We leave open the computational complexity of 3-IS and 4-IS.

Proposition 1. 2-IS *can be solved in polynomial time.*

Proof. We assume that there is at least one negative edge. Otherwise, the grand coalition is Nash-stable. The algorithm goes as follows:

> Start with any bipartition. Move nodes with incident negative edges so that they have a negative edge to the other coalition. In each of the two coalitions, contract all nodes with negative incident edges into a single node and call the contracted nodes s and t. For any other node the new edge weights to s and

t are the sum of the original edge weights to the corresponding contracted nodes. Now (ignoring all edges between s and t) compute a min cut between s and t via a max flow algorithm and assign the nodes accordingly.

After the first stage, all nodes that we are about to contract have a negative edge to the other coalition. So they are not allowed to join the other coalition. This property is preserved by contraction. Afterwards, the flow algorithm operates only on positive edges and computes a global minimum cut between s and t. Thus, the cut also maximizes the total happiness of all non-contracted nodes, so none of these nodes has an incentive to switch coalitions. All performed steps of the algorithm can be done in polynomial time. □

Next we show that IS is PLS-complete[1] when we do not impose a restriction on the number of coalitions in the problem definitions as we did for 2-IS.
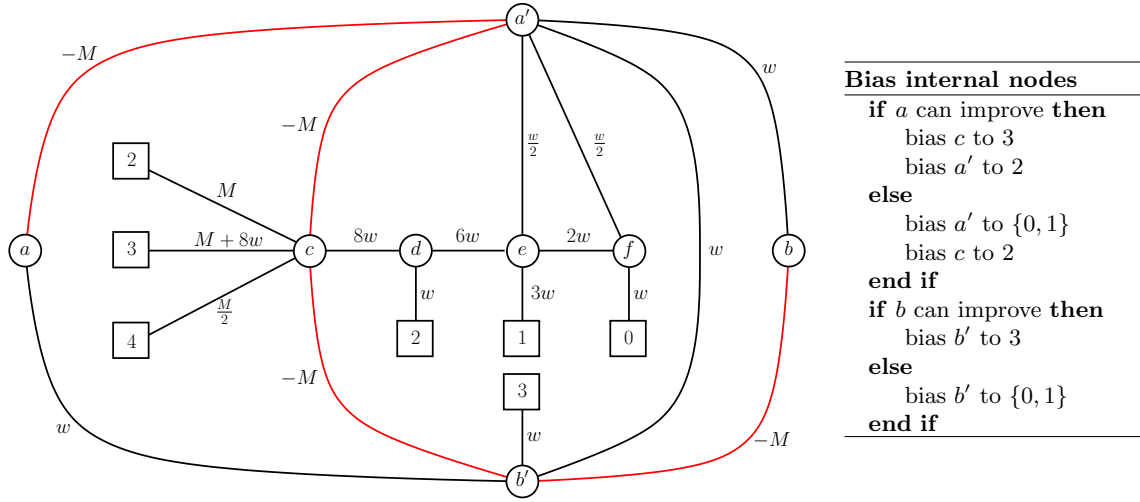
Theorem 3. IS *is PLS-complete.*

Proof. We start with an instance of OneEnemyPartyAffiliation*. The instance has the property that no player is ever indifferent between the two coalitions that make up stable outcomes. We add five supernodes which are connected by a complete graph of sufficiently large negative edges. This enforces that in any stable outcome the supernodes are in different coalitions, say 0, 1, 2, 3, 4. The supernodes are used to restrict which coalition a node can be in in a stable outcome. This is achieved by having large positive edges of equal weight to the corresponding supernodes. All original nodes of the OneEnemyPartyAffiliation* instance are restricted to be 0 or 1.

We now show how to simulate a negative edge of OneEnemyPartyAffiliation* by an IS-gadget. To do so, we replace a negative edge (a, b) of weight $-w$ with the gadget in Figure 5. Nodes a and b are original nodes and restricted to $\{0, 1\}$, node a' is restricted to $\{0, 1, 2\}$, node b' is restricted to $\{0, 1, 3\}$, node c is restricted to $\{2, 3, 4\}$, node d is restricted to $\{2, 4\}$, node e is restricted to $\{1, 2\}$, node f is restricted to $\{0, 2\}$. As depicted in the gadget, nodes b', c, d, e, and f have additional offsets.

Coalitions $2, 3$ and 4 are only used locally within the gadget. The pseudocode next to the gadget describes how the internal nodes of the gadget are biased. Here, checking whether a node can improve is w.r.t. her *original* neighborhood. We use "look at" and "bias" as defined in the following lemma and definition, which are analogous to those in [17, 25]. In particular, we check if a node can improve by looking at all nodes in her original neighborhood.

Lemma 8. *For any polynomial-time computable function $f : \{0, 1\}^k \mapsto \{0, 1, 2, 3\}^m$ one can construct a graph $G_f = (V_f, E_f, w)$ having the following properties: (i) there exist $s_1, \ldots, s_k, t_1, \ldots, t_m \in V_f$, (ii) all edges $e \in E_f$ are positive, (iii) $f(s_1, \ldots, s_k) = (t_1, \ldots, t_m)$ in any stable solution of the hedonic game defined by G_f.*

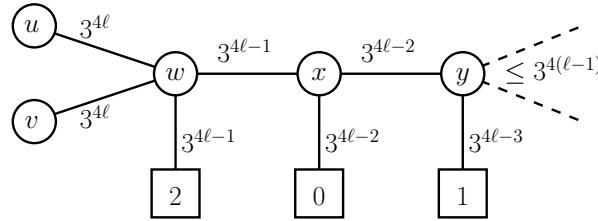[1] The version of Theorem 3 that appeared in [20] missed a special case that is dealt with here.

Fig. 5. Gadget to replace negative edges

Proof. We first show how to construct a graph $G_{f'}$, which implements a function $f' : \{0,1\}^k \mapsto \{0,1\}^{4m}$. This part is similar to the proof of Lemma 1. Afterwards, we show how to augment $G_{f'}$ to implement $f : \{0,1\}^k \mapsto \{0,1,2,3\}^m$

It is well known that for any polynomial computable function $f' : \{0,1\}^k \mapsto \{0,1\}^{4m}$ one can construct a circuit C with polynomial many gates that implements this function [31, Theorem 9.30]. Clearly, we can also restrict C to NOR gates with fan-in and fan-out at most 2. Organize the gates in levels according to their distance to C's output; output gates of C are at level 1.

We replace each gate g_i at level ℓ with the gadget in Figure 6. Nodes u, v are inputs and y is the output of the gate. Nodes u, v, y are restricted to $\{0, 1\}$. Nodes w and x are internal to the gate and restricted to $w \in \{1, 2\}$ and $x \in \{0, 2\}$, respectively. By construction of the NOR gate, we have that in any Nash-stable solution, $y = 1$ if and only if $u = v = 0$. In other words $y = NOR(u, v)$.

Fig. 6. NOR gate without negative edges

If u (or v) is an input of the circuit then we connect u to the corresponding input s-node by an edge of weight $3^{4\ell+1}$. To connect the output t-nodes, we need to augment

$G_{f'}$ in order to allow for the extended range of the function $f : \{0,1\}^k \mapsto \{0,1,2,3\}^m$. For each output node t_i we will use 4 outputs of $G_{f'}$.

For each of this 4 outputs, we first change the domain by using a slightly modified NOR-gate. Observe, that by changing the offsets and restrictions of nodes w, x, and y in a NOR-gate, we can change the domain of the NOR gate to any two distinct values in $\{0,1,2,3,4\}$. E.g., if we want $y \in \{2,4\}$ and $y = 2$ if and only if $u = v = 0$, then we can change the restrictions $w \in \{1,3\}$, $x \in \{2,3\}$, and $y \in \{2,4\}$; and the offset of w to 3, x to 2, and y to 4. Using this idea, we first change the domain of the 4 outputs of C to $\{0,4\}$, $\{1,4\}$, $\{2,4\}$, and $\{3,4\}$, respectively. The 4 modified outputs are then all connected to the output node t_i (which we restrict to $\{0,1,2,3\}$) with edges of weight 1. By the right choice of f', in particular by ensuring that f' forces exactly one of the modified outputs being $\neq 4$, we can implement any function f.

The claim follows since our construction fulfils properties (i), (ii) and (iii). $\qquad\square$

Definition 6. *For a polynomial-time computable function $f : \{0,1\}^k \mapsto \{0,1,2,3\}^m$ we say that G_f as constructed in Lemma 8 is a graph that* looks at $s_1, \ldots, s_k \in V_f$ *and* biases $t_1, \ldots, t_m \in V_f$ *according to the function f.*

Recall that the instance of OneEnemyPartyAffiliation* has the property that no player is ever indifferent between the two coalitions that make up stable outcomes. By scaling edge weights we can implement the "look at" required to bias the internal nodes of the gadget without affecting their original preferences.

We say that node a is *locked* by the gadget if $a = 1$ and $a' = 0$ or $a = 0$ and $a' = 1$. Node b is said to be locked accordingly. The following three lemmas describe the operation of the gadget. All three lemmas should be read with the implicit clause: *If the internal nodes (a', b', c, d, e, f) of Figure 5 are stable.* Let $\neg u$ denote the complement of u over $\{0,1\}$.

Lemma 9. *Node c is either in coalition 2 or 3, while nodes d, e and f are in coalition 2.*

Proof. We start by showing that $c \in \{2,3\}$. By way of contradiction assume $c = 4$. Then $c = 4 \Rightarrow d = 4 \Rightarrow e = 1 \Rightarrow f = 0$. Now a' strictly prefers coalitions 0 and 1 to coalition 2 and is only blocked by a from entering one of those coalitions. Thus $a' \in \{0,1\}$. Together with $b' \in \{0,1,3\}$ this directly implies that c can improve by choosing $c = 2$, contradicting our assumption. Thus $c \in \{2,3\}$.

From $c \neq 4$ we immediately get $d = 2 \Rightarrow e = 2 \Rightarrow f = 2$, which completes the proof the lemma. $\qquad\square$

Lemma 10. *If neither a nor b can improve then a and b are locked by the gadget.*

Proof. Since neither a nor b can improve, a' and b' are biased to $\{0,1\}$ and c is biased to 2. If $c = 2$ then the bias on a' assures $a' = \neg a$. So b' has an edge of weight w to both 0 and 1. Together with the bias this implies $b' = \neg b$. If $c = 3$ then the bias on b' assures $b' = \neg b$. So a' has an edge of weight w to both 0 and 1. Together with the

bias this implies $a' = \neg a$. So in both cases $a' = \neg a$ and $b' = \neg b$. The claim follows. □

Lemma 11. *If a or b (or both) can improve then one improving node is not locked while the other node is locked by the gadget. Moreover, if a (resp. b) is not locked by the gadget then $b' = \neg b$ (resp. $a' = \neg a$).*

Proof. We consider three cases: (i) only a can improve, (ii) only b can improve, (iii) a and b can improve.

Case (i) (only a): Here c is biased to 3, a' is biased to 2, and b' is biased to $\{0, 1\}$. First assume $c = 2$. This enforces $a' = \neg a$ which together with the bias implies $b' = \neg b$. But then the bias on c gives $c = 3$, a contradiction. Thus $c = 3$, which enforces $b' = \neg b$ and with the bias implies $a' = 2$. So a is not locked and b is locked.

Case (ii) (only b): Here c is biased to 2, a' is biased to $\{0, 1\}$, and b' is biased to 3. First assume $c = 3$. This enforces $b' = \neg b$ which together with the bias implies $a' = \neg a$. But then the bias on c gives $c = 2$, a contradiction. Thus $c = 2$, which enforces $a' = \neg a$ and with the bias implies $b' = 3$. So a is locked and b is not locked.

Case (iii) (a and b): Here c is biased to 3, a' is biased to 2, and b' is biased to 3. If $c = 2$ then this enforces $a' = \neg a$, which together with the bias implies $b' = 3$. So in this case a is locked and b is not locked. If $c = 3$ then this enforces $b' = \neg b$, which together with the bias implies $a' = 2$. So in this case a is not locked and b is locked.

In every case both claims of the lemma are fulfilled. □

To complete the proof we show that a stable outcome of the IS instance is also a stable outcome for the ONEENEMYPARTYAFFILIATION* instance. Suppose the contrary. Then there must exist an original node which is stable for IS but not for ONEENEMYPARTYAFFILIATION*. Clearly such a node must be the node a or b for some gadget. So either a or b (or both) can improve. But then by the first statement in Lemma 11 one of the improving nodes is unlocked, say a. Since a was only incident to one negative edge in the ONEENEMYPARTYAFFILIATION* instance, a cannot be locked by any other gadget. Moreover, by the second statement in Lemma 11, a is now connected in the gadget by a positive edge to the node b' and $b' = \neg b$. On the one hand, if $a = b$ then the original edge (a, b) contributes $-w$ to a's utility while now a receives 0 from the edge (a, b'). On the other hand, if $a \neq b$ then the corresponding utility contributions are 0 and w. So if a changes strategy then the difference in her utility w.r.t. b is the same in both problems, since we just shifted the utility of node a w.r.t. b by w. So a is also not stable for IS, a contradiction. This finishes the proof of Theorem 3. □

5   Other veto-based stability concepts

In IS a single player can veto against others joining her coalition but there is no restriction on leaving a coalition. The following proposition shows that adding certain

leaving conditions yields polynomial-time convergence from the all-singleton partition.

Proposition 2. *Any problem in column 3 of Figure 2 can be solved in polynomial time provided that the leaving condition requires that the leaving node has at least one negative edge within the coalition. In particular this hold for the problems in cells 3B, 3C, and 3D.*

Proof. We use local improvements starting from the set of singleton coalitions. Then a player can make at most one improving step, since all edges in resulting non-singleton coalitions will be positive because of the veto-in restriction, and so no player can leave such a coalition. Hence we arrive at a stable outcome in at most $|V|$ improving steps. □

Interestingly, requiring veto-out feasibility is already enough for polynomial-time convergence even if we have no restriction on the entering condition. This stands in contrast to Theorem 3.

Proposition 3. *All problems in row C of Figure 2 can be solved in polynomial time by local improvements using at most $2|V|$ improving steps.*

Proof. To get a running time of $2|V|$ (rather than $O(|V|^2)$) we restrict players from joining a non-empty coalition to which they have no positive edge. This ensures that whenever a player joins a non-empty coalition then this player (and all players to which she is connected by a positive edge in the coalition) will never move again. Moreover, a player can only start a new coalition once. It follows that each player can make at most two strategy changes. In total we have at most $2|V|$ local improvements. □

6   sumCIS

Next we study SUMCIS, where a deviating player's total weight to the new coalition is non-negative, and to the old coalition is non-positive. Even though deviations are very restricted here, it is PLS-complete to compute a stable outcome.

Theorem 4. SUMCIS *is PLS-complete.*

Proof. We reduce from LOCALMAXCUT. Consider an arbitrary instance of LOCALMAXCUT with only integer edge weights. Recall that such an instance can be cast as an instance of PARTYAFFILIATION by negating the weights of the edges. Let $G = (V, E, w)$ represent the PARTYAFFILIATION instance. For each player $i \in V$ let σ_i be the total weight of edges incident to player i, i.e. $\sigma_i = \sum_{(i,j) \in E} w_{(i,j)}$. Observe that σ_i is a negative integer. We augment G by introducing two new players, called *supernodes*. Every player $i \in V$ has an edge of weight $\frac{-\sigma_i}{2} + \frac{1}{4}$ to each supernode. The

two supernodes are connected by an edge of weight $-M$ where M is sufficiently large (i.e., $M > \sum_{i \in V}(\frac{-\sigma_i}{2} + \frac{1}{4})$). The resulting graph G' represents our SUMCIS instance.

Consider a stable outcome of the SUMCIS instance G'. By the choice of M the two supernodes will be in different coalitions. Now consider any player $i \in V$. If i is not in a coalition with one of the supernodes, then i's payoff is negative. On the other hand joining the coalition of one of the supernodes yields positive payoff, since $2(\frac{-\sigma_i}{2} + \frac{1}{4}) + \sigma_i > 0$. Thus, each player $i \in V$ will be in a coalition with one of the supernodes. So our outcome partitions V into two partitions, say V_1, V_2.

It remains to show that any stable outcome for the SUMCIS instance is also a local optimum for the PARTYAFFILIATION instance. Assume that the outcome of the SUMCIS instance is stable but in the corresponding outcome of PARTYAFFILIATION instance there exists a player i which can improve by joining the other coalition. W.l.o.g. assume $i \in V_1$. Then, $\sum_{s \in V_1} w_{(i,s)} < \sum_{s \in V_2} w_{(i,s)}$. With $\sigma_i = \sum_{s \in V} w_{(i,s)}$ and since σ_i is integer, we get

$$\sum_{s \in V_1} w_{(i,s)} \leq \frac{\sigma_i}{2} - \frac{1}{2} < \frac{\sigma_i}{2} < \frac{\sigma_i}{2} + \frac{1}{2} \leq \sum_{s \in V_2} w_{(i,s)}.$$

It follows that in the SUMCIS instance, player i's payoff is negative in her current coalition V_1 whereas joining V_2 would yield positive payoff. This contradicts our assumption that we are in a stable outcome of the SUMCIS instance. The claim follows. $\square$

7 Voting-based deviations

In this section we study the complexity of computing stable outcomes under various voting-based stability requirements. We start by showing PLS-hardness for the case that a deviating player needs a T_{in} majority in the target coalition but there is no restriction on leaving coalitions.

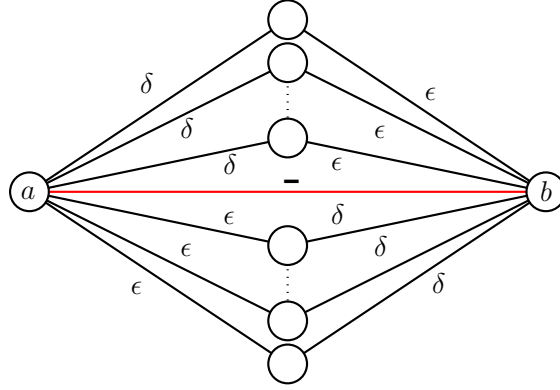**Theorem 5.** VOTEIN *is PLS-complete for any voting threshold* $0 \leq T_{in} < 1$.

Proof. We reduce from ONEENEMYPARTYAFFILIATION* represented by an edge-weighted graph $G = (V, E, w)$. Let $\Delta(G)$ be the maximum degree of a node in G. Recall that no player is ever indifferent between the two coalitions.

First observe that the case $T_{in} > \frac{\Delta(G)-1}{\Delta(G)}$ is exactly the same as IS (for which we show hardness in Theorem 3), since in this case one negative edge is enough to veto a player joining a coalition. In the following we assume $T_{in} \leq \frac{\Delta(G)-1}{\Delta(G)}$.

We augment G as follows:

For every negative edge (a, b) in G we introduce $2\Delta(G) - 2$ new nodes, called *followers*, and connect them with a and b as shown in the Figure 7. Both, a and b, get $\Delta(G)-1$ followers and have a δ edge to each of them. Moreover, the followers have also an edge of weight ε to the other node. Here $0 < \varepsilon < \delta$ and δ is small enough so

Fig. 7. Gadget used for showing that VOTEIN is PLS-complete. The gadget augments negative edges with followers that ensure that there is always a T_{in}-majority when a player enters a coalition.

that the player preferences of the original players (a and b) are still determined only by the original edges. In a stable outcome the followers will be in the same coalition as their "leader", i.e., the node to which they have a δ edge. The followers make sure that their is always a T_{in}-majority for entering a coalition. In other words, in a stable outcome of the VOTEIN instance, the voting doesn't impose any restrictions.

To ensure that any stable outcome for the VOTEIN instance has only two coalitions we further augment G by introducing two new players, called *supernodes*. Every player $i \in V$ has an edge of weight $W > \sum_{e \in E} |w_e|$ to each of the supernodes. The two supernodes are connected by an edge of weight $-M$, where $M > |V| \cdot W$. This enforces that the two supernodes are in a different coalition in any stable outcome. Moreover, by the choice of W, each player in V will be in a coalition with one of the supernodes. The fact that edges to supernodes have all the same weight directly implies that a stable outcome for the VOTEIN instance is also a stable outcome for the ONEENEMYPARTYAFFILIATION* instance. The claim follows. □

In contrast to VOTEIN, VOTEOUT is conceptually different. In VOTEOUT a coalition of two players connected by a positive edge is vote-out stable. This makes it hard to restrict the number of coaltions. Doing this is probably the key for proving PLS-hardness also for VOTEOUT. For the following theorem we consider a version of VOTEOUT where the number of coalitions are restricted by the problem. Let k-VOTEOUT be the problem of computing a vote-out stable outcome when at most k coalitions are allowed. Observe that for any $k \geq 2$ such a vote-out stable outcome exists and that local improvements starting from any k-partition converge to such a stable outcome.

Theorem 6. k-VOTEOUT *is PLS-complete for any voting threshold* $0 \leq T_{out} < 1$ *and any* $k \geq 2$.

Proof. Our reduction is from ONEENEMYPARTYAFFILIATION, but we first reduce to the intermediate problem ONEENEMYNASHSTABLE, which is a restricted version

of NASHSTABLE where each player is only incident to at most one negative edge. Consider an instance of ONEENEMYPARTYAFFILIATION which is represented as an edge-weighted graph $G = (V, E, w)$. We augment G with two supernodes in exactly the same way as in Theorem 5. This ensures that any stable outcome of the ONEENEMYNASHSTABLE instance uses only two coalitions and thus is also a stable outcome for the ONEENEMYPARTYAFFILIATION instance. Hence, ONEENEMYNASHSTABLE is PLS-complete.

We now reduce from ONEENEMYNASHSTABLE to k-VOTEOUT. Let G be the graph corresponding to an instance of ONEENEMYNASHSTABLE. Let $\Delta(G)$ be the maximum degree of a node in G. We augment G as follows: We introduce $s \cdot k \cdot \Delta(G)$ new nodes where s is an integer satisfying $s \geq \frac{T_{out}}{1-T_{out}}$. Those nodes are organized in $s \cdot \Delta(G)$ complete graphs of k nodes each. All the edges in the complete graphs have weight $-M$ where M is sufficiently large ($M > |V| \cdot \Delta(G) \cdot \varepsilon$ will do). Moreover, we connect every original node $u \in V$ to every new node with an edge of weight $-\varepsilon$, where $\varepsilon > 0$.

By the choice of M and since at most k coalitions are allowed, in any stable solution there will be one node from each complete graph in each of the k coalitions. This shifts the utility of each player $i \in V$ with respect to each coalition by $-s \cdot \Delta(G) \cdot \varepsilon$. Moreover, every original node has at least $s \cdot \Delta(G)$ negative edges to each coalition. Since each node is incident to at most $\Delta(G)$ positive edges, it follows that the fraction of negative edges to each coalition is at least $\frac{s}{s+1} \geq T_{out}$. Thus, in every stable outcome all nodes $u \in V$ have a T_{out}-majority for leaving their coalition. This implies that in the corresponding outcome of the ONEENEMYNASHSTABLE instance, no player can improve her utility by joining one of the k coalitions used in k-VOTEOUT. Moreover, in every stable outcome the utility of each node $u \in V$ with respect to the set of original nodes V is non-negative, since u has at most one negative incident edge in the ONEENEMYNASHSTABLE instance and $k \geq 2$. It follows that a stable outcome for the k-VOTEOUT instance is also a stable outcome for the ONEENEMYNASHSTABLE instance. The claim follows. $\square$

It is an interesting open problem whether PLS-completeness also holds if the restriction on the number of allowed coalitions is dropped. Can we construct a gadget that imposes this restriction without restricting the problem a priori?

Since VOTEIN and a restricted version of VOTEOUT are PLS-complete it's interesting to study the combination of both problems. What happens if we require vote-in stability *and* vote-out stability? With a mild assumption on the voting thresholds T_{in}, T_{out}, we establish:

Theorem 7. *For any instance of* VOTEINOUT *with voting thresholds* $T_{in}, T_{out} \geq \frac{1}{2}$ *and* $T_{in} + T_{out} > 1$, *local improvements converge in* $\mathcal{O}(|E|)$ *steps.*

Proof. For any outcome p define a potential function $\Phi(p) = \Phi^+(p) - \Phi^-(p)$, where $\Phi^+(p)$ (resp. $\Phi^-(p)$) is the number of positive (resp. negative) internal edges, i.e. edges not crossing coalition boundaries. Consider a local improvement of some player

i from coalition $p(i)$ to $p'(i)$. Since $T_{out} \geq \frac{1}{2}$, player i has at least as many negative as positive edges to $p(i)$. Likewise since $T_{in} \geq \frac{1}{2}$, player i has at least as many positive as negative edges to $p'(i)$. So $\Phi(p)$ cannot decrease by a local improvement. Moreover, since $T_{in} + T_{out} > 1$, one of the threshold inequalities must be strict, which implies $\Phi(p') > \Phi(p)$. The claim follows since $-|E| \leq \Phi(p) \leq |E|$ and $\Phi(p)$ is integer.   □

Without the assumption on the voting thresholds from Theorem 7, the complexity of computing stable outcomes is an interesting open problem, in particular the case $T_{in} = T_{out} = 1/2$.

8   Open problems

In this paper, we studied the computational complexity of finding stable outcomes in hedonic games. We show that NASHSTABLE is PLS-complete. On the other hand we show that CIS, that is finding a stable outcome where any member of a coalition can block (veto) a player from leaving or joining, can be solved in polynomial time. For the case that a player can only block a player from joining, we show that the corresponding problem IS is PLS-complete (Theorem 3). Our reduction to IS uses five coalitions. On the other hand, 2-IS, where the number of coalitions is restricted to two, is solvable in polynomial time (Proposition 1). This leaves open the complexity of 3-IS and 4-IS, where the number of coalitions is restricted to three or four, respectively.

We then study cases where members of a coalition can vote on whether to allow a player to leave or join a coalition. The problem VOTEIN is parameterized by a voting threshold, $T_{in} \in [0, 1]$. IS can be seen as VOTEIN with $T_{in} = 1$. Theorem 5 shows hardness for $0 \leq T_{in} < 1$, so in fact we show that VOTEIN is PLS-complete for all voting thresholds. In contrast, we show that the case of VOTEOUT with $T_{out} = 1$ is polynomial-time solvable (Proposition 3). This suggests that VOTEOUT is conceptually different from VOTEIN. Indeed, it seems difficult to restrict the coalitions in this case. We do show that k-VOTEOUT, where we restrict the outcome to have at most k coalitions, is PLS-complete for $0 \leq T_{out} < 1$, but we leave the complexity of VOTEOUT as an interesting open problem.

On the positive side, we show that local improvements converge in polynomial time in the case of requiring *both* vote-in- and vote-out- stability with $T_{in}, T_{out} \geq 0.5$ and $T_{in} + T_{out} > 1$. We leave open the interesting case of VOTEINOUT with voting thresholds that do not satisfy $T_{in}, T_{out} \geq \frac{1}{2}$ and $T_{in} + T_{out} > 1$. We also leave open the case of finding an outcome that is vote-in and sum-out stable.

Elsässer and Tscheuschner [17] showed that local max cut is PLS-complete even when the input graph has degree at most five. In contrast, Poljak [29] gives a polynomial-time algorithm for graphs with degree at most three. It would be interesting to study degree restrictions for additively-separable hedonic games.

Bibliography

[1] E. H. L. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. Wiley-Interscience, 1997.

[2] H. Aziz and R. Savani. Hedonic games. In *Handbook of Computational Social Choice*. Cambridge University Press, 2015. In press.

[3] H. Aziz, F. Brandt, and H. G. Seedig. Stable partitions in additively separable hedonic games. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 183–190, 2011.

[4] M.-F. Balcan, A. Blum, and Y. Mansour. Improved equilibria via public service advertising. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 728–737, 2009.

[5] C. Ballester. NP-completeness in hedonic games. *Games and Economic Behavior*, 49(1):1–30, 2004.

[6] A. Bhalgat, T. Chakraborty, and S. Khanna. Approximating pure Nash equilibrium in cut, party affiliation and satisfiability games. In *Proc. of the ACM Conference in Electronic Commerce (EC)*, pages 132–146, 2010.

[7] A. Bogomolnaia and M. O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.

[8] S. Branzei and K. Larson. Coalitional affinity games and the stability gap. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1319–1320, 2009.

[9] N. Burani and W. S. Zwicker. Coalition formation games with separable preferences. *Mathematical Social Sciences*, 45(1):27–52, 2003.

[10] K. Cechlárová. Stable partition problem. In *Encyclopedia of Algorithms*. Springer, 2008.

[11] G. Christodoulou, V. S. Mirrokni, and A. Sidiropoulos. Convergence and approximation in potential games. In *Proc. of the Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 349–360, 2006.

[12] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 12(2):257–266, 1994.

[13] J. H. Dreze and J. Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica*, 48(4):987–1003, 1980.

[14] E. Elkind and D. Pasechnik. Computing the nucleolus of weighted voting games. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 327–335, 2009.

[15] E. Elkind and M. Wooldridge. Hedonic coalition nets. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 417–424, 2009.

[16] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. On the computational complexity of weighted voting games. *Ann. Math. Artif. Intell.*, 56 (2):109–131, 2009.

[17] R. Elsässer and T. Tscheuschner. Settling the complexity of local max-cut (almost) completely. In *Proc. of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 171–182, 2011.

[18] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The Complexity of Pure Nash Equilibria. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004.

[19] M. Gairing and R. Savani. Computing stable outcomes in hedonic games. In *Proc. of the International Symposium on Algorithmic Game Theory (SAGT)*, 2010.

[20] M. Gairing and R. Savani. Computing stable outcomes in hedonic games with voting-based deviations. In *Proc. of the International conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 559–566, 2011.

[21] J. Greenberg. Coalition structures. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume II. Elsevier, 1994.

[22] S. Ieong and Y. Shoham. Marginal contribution nets: A compact representation scheme for coalitional games. In *Proc. of the ACM Conference on Electronic Commerce (EC)*, page 193, 2005.

[23] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How Easy is Local Search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[24] M. W. Krentel. Structure in Locally Optimal Solutions. In *Proc. of the Symposium on Foundations of Computer Science (FOCS)*, pages 216–221, 1989.

[25] B. Monien and T. Tscheuschner. On the power of nodes of degree four in the local max-cut problem. In *Proc. of the International Conference on Algorithms and Complexity (CIAC)*, 2010.

[26] B. Monien, D. Dumrauf, and T. Tscheuschner. Local search: Simple, successful, but sometimes sluggish. In *Proc. of the International Colloquium on Automata, Languages, and Programming (ICALP)*, 2010.

[27] J. B. Orlin, A. P. Punnen, and A. S. Schulz. Approximate local search in combinatorial optimization. *SIAM Journal on Computing*, 33(5):1201–1214, 2004.

[28] D. Peters and E. Elkind. Simple causes of complexity in hedonic games. In *Proc. of the International Conference on Artificial Intelligence (IJCAI)*, pages 617–623, 2015.

[29] S. Poljak. Integer linear programs and local search for max-cut. *SIAM Journal on Computing*, 21(3):450–465, 1995.

[30] A. A. Schäffer and M. Yannakakis. Simple Local Search Problems that are Hard to Solve. *SIAM Journal of Computing*, 20(1):56–87, 1991.

[31] M. Sipser. *Introduction to the Theory of Computation*. Thomson, 2006.

[32] S. C. Sung and D. Dimitrov. On myopic stability concepts for hedonic games. *Theory and Decision*, 62, 2007.

[33] S. C. Sung and D. Dimitrov. Computational complexity in additive hedonic games. *European Journal of Operational Research*, 203(3):635–639, 2010.

[34] M. Yannakakis. Equilibria, fixed points, and complexity classes. In *Proc. of the International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 19–38, 2008.